Coursework – CN7023

Artificial Intelligence & Machine Vision

Siddharth Dholu – u2165171
MSc. Artificial Intelligence
University of East London, UK
siddharthdholu@gmail.com

Abstract – We humans are limited to some extension to extract the feature manually so in this case we need help of Convolutional Neural Networks (CNN) where this problem could be solved. So, here we have utilized CNN for image classification to identify the correct one. As in CNN, I will use the important features from the top layer of the CNN for grouping; in any case, those elements may not contain sufficient helpful data to accurately anticipate a picture. Sometimes, features from the lower layer convey more discriminative power than those from the top. Hence, applying features from a particular layer just to grouping is by all accounts a cycle that doesn't use took in CNN's potential discriminant ability to its full degree. On account of this property, we are needing combination of important features from numerous layers. We need to make a model with numerous layers that will identify or recognize the images. I need to complete our model by utilizing the concepts of CNN and Natural Images dataset which is provided by our professionals. The target of my work is to learn and basically apply the ideas of Convolutional Neural Network.

Keywords – CNN (Convolutional Neural Networks), dataset, image classification, layer, filter, feature, model, padding, stride, softmax, adam, network, AI (Artificial Intelligence)

I. INTRODUCTION

CNN becomes perhaps the most engaging methodologies as of late and has been an extreme component in an assortment of ongoing achievement and testing application connected with Artificial Intelligence applications, for example, challenge image classification and face detection or recognition. Consequently, we think about CNN as our model for our difficult errands of picture characterization. We use CNN for division and classification of the pictures in scholar and deals. We use picture acknowledgment in various regions for instance mechanized picture association, stock photography, face acknowledgment, and numerous other related works.

CNN is nothing but, deep artificial neural networks. We use CNN to group the pictures, or we can say images, bunch them by similitude (photograph search), and perform object acknowledgment inside scenes. It tends to be utilized to recognize faces, individual, road signs, growths, platypuses and numerous different parts of visual information. The convolutional layer is the centre structure square of a CNN. The layer's boundaries comprise of a bunch of learnable channels (or pieces) which have a little open field yet stretch out through the full profundity of the info volume. During the forward pass, each channel is convolved across the width and level of the info volume, processing the speck item, and delivering a 2-layered initiation guide of that channel. Thus, the organization finds out about the channels. The channel actuates when they see some sort of element at some spatial situation in the information. Then, at that point, the enactment maps are taken care of into a down sampling layer, and like convolutions, this strategy is applied each fix in turn. CNN has additionally completely associated layer that arranges yield with one label for each node.

II. SIMULATIONS

a) Natural Images Dataset

We utilize this dataset for deep machine learning to train it for image predictions. This Natural Images dataset is the contribution of Prasun Roy, Saumik Bhattacharya, Subhankar Ghosh and Umapada Pal. This dataset contains 6,899 images from 8 distinct classes compiled from various sources. The classes include airplane, car, cat, dog, flower, fruit, motorbike and person. The images are various size pixel. The dataset comprises of 80 percent for training and 20 percent testing to build the model. An information base for individuals need to take a stab at learning procedures and example acknowledgment strategies on true information while spending insignificant endeavors on preprocessing and designing. We will involve this data set in our trial.

The dataset is available on internet in Kaggle's database. The link is https://www.kaggle.com/datasets/prasunroy/natural-images?resource=download and for program we have created link https://download1336.mediafire.com/n4a6elomuzng/90z98y3gjtim388/natural_images.tgz. It is in compressed format in .zip extension which we need to extract it and inside we will find folders name "natural_images" which contains sub folders inside it as name of classes.

The compressed database size is 173 MB and after extraction it will be 179 MB. Each sub folder has different number of images.

	A	B
1	Classes (Sub Folder)	Files Count
2	airplane	727
3	car	968
4	cat	885
5	dog	702
6	flower	843
7	fruit	1000
8	motorbike	788
9	person	986
10		
11	TOTAL	6899

Fig. 1: Directory tree & Sub directory summary

As in figure 1 we can see that, the database has 8 sub folder (classes/labels) and each sub folder has different number of files like airplane has 727, car has 968, and so on. In last we have total 6899 images.

Figure 2 shows the raw sample (image) of each class. All samples are having different dimensions and sizes. So, we need to do operation on each image to make it in same dimension.

Fig. 2: Sample of dataset images

b) Encoding Dataset

We have downloaded the dataset from net by passing url into keras function which is famous library for image classification to import into our IDE. We extracted compressed file into our local host machine by passing untar true in function as perameter. In our case, we have used Macintosh system to run this experiment. We have used Google Colab (https://colab.research.google.com/) for building the CNN model which is online platform that allows us to write the code directly on cloud with using Python language. As we know, Python is best language for machine learning and open-source language.

Python has many libraries for loading the data into its environment. In this case, we have used TensorFlow, CV2 and PIL libraries to make the operation on dataset.

TensorFlow provides the method to import the image dataset from local machine to python IDE. Once we successfully imported the dataset, we need to check the images' dimensions which is most important factor for model building. As this dataset has various dimensions of images, we have resized each image into 180x180 dimension with the help of CV2 library and store newly created dataset into new array which contains pixel values between 0 to 255 of each RGB channel. As we can see in figure 2 that images have been resized into same dimensions.

Fig. 3: Samples after decoding

In figure 3 all images have been decoded and converted into same dimensions. Now it is ready for training and testing.

This dataset contains eight classes, so we have grouped the images by its classes and created dictionary (map) for future use to build the model. Alongside we have assigned label to each image with the help dictionary into arrays. Finally, we converted those arrays to NumPy array. Because our CNN model needs NumPy array data to train and test the data.

Ultimately, we have split the data into 80 – 20, where 80 percent for training and 30 percent for testing to create CNN model.

```
# Importing library for path related operation
import pathlib

# Natural Images URL
dataset_url = 'https://download1336.mediafire.com/n4a6elomuzng/90z98y3gjtim388/natural_images.tgz'

# Importing dataset from net
data_dir = tf.keras.utils.get_file('natural_images', origin=dataset_url, untar=True)
data_dir = pathlib.Path(data_dir)

batch_size = 32
img_height = 180
img_width = 180

# Spliting dataset into training dataset
train_ds = tf.keras.utils.image_dataset_from_directory(
      data_dir,
      validation_split=0.2,
      subset="training",
      seed=123,
      image_size=(img_height, img_width),
      batch_size=batch_size)

# Spliting dataset into validating (testing) dataset
val_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

Fig. 4: Snippet code for dataset importing and encoding

We have imported dataset from net using url with above configurations as in figure 4 in python Google Colab. This is how we encoded the data for model to train and test it.

We have given batch size of 32 and image size 180x180 to resize the images into same dimensions. We split the data into 20 – 80 percent.

c) Network Architecture

Deep learning with neural network, it understands that quite possibly the most administered deep learning procedure is the Convolutional Neural Network. We plan Convolutional Neural Network to perceive visual examples straightforwardly from pixel pictures with insignificant preprocessing. Practically all CNN structures follow similar general plan standards of progressively applying convolutional layers to the info, intermittently down sampling (Max pooling) the spatial aspects while expanding the quantity of element maps. Besides, there are likewise completely associated layers, initiation capacities and misfortune work (e.g., cross entropy or softmax). Be that as it may, among every one of the tasks of CNN, convolutional layers, pooling layers, and completely associated layers are the main ones. Accordingly, we will rapidly present these layers prior to introducing our proposed model.

The first layer is convolutional layer where it can extract important features itself from the image. The pixels relate to linear values and nearest pixels, it allows to protect the connection between different dots of an image. It is also sifting the pixel by pixel with more modest pixel channel to diminish the size of the image with taking care that without losing any connection between the pixels. At the point when we apply convolution to a 7x7 picture by utilizing a channel of size 3x3 with 1x1 step (1-pixel shift at each progression), we will wind

up having a 5x5 result.

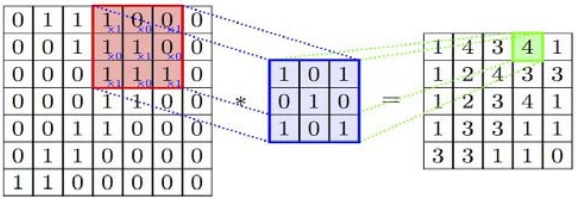

Fig. 5: Convolution Operation

While developing convolutional neural network, it is very common to embed the layers of pooling after each convolution layer, so we can diminish the spatial portrayal size. This layer will be decreased the boundary counts of pixels of image. In addition to that, pooling layer helps from the overfitting data which will solve overfitting issue. We can choose the size of pooling layer that how much we want to decrease the boundaries at most extreme, normal or total numbers inside pixels. As figure 6 shows that max pooling and normal pooling tasks.

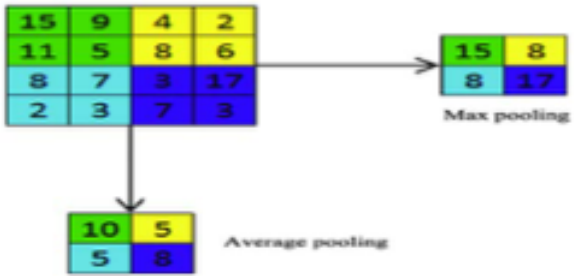

Fig. 6: Max pooling & Average pooling operation

An entirely connected network is like any engineering where each dot is connected to each other to make the decision of connected and impact of every boundary on the marks. We can incomprehensibly diminish the time-space intricacy by utilizing the convolution and pooling layers. We can develop a completely associated network eventually to predict our images.

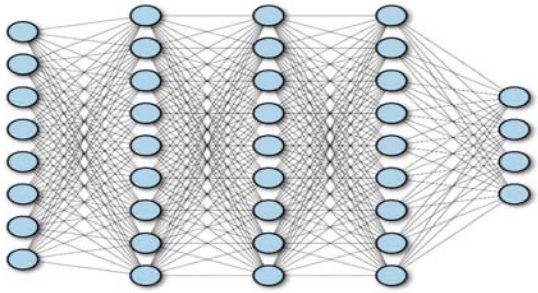

Fig. 7: Fully connected layer

Fig.8 shows the outline look of our proposed CNN model. It is especially like the other image classification structures [5,6,7,8] however has changed in the quantity of filters, neurons and actuation capacities for better execution. We can partition our model into six groupings of layers.

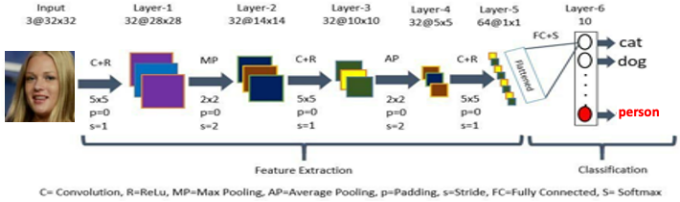

Fig. 8: The architecture of our proposed CNN

d) Explanation of Algorithm

A straightforward convolutional network is a series of layers. The layer changes into one volume of initiations to one more through a differentiable capacity. We utilize three main kinds of layers to construct network design. They are a convolutional layer, pooling layer, and completely associated layer. We will stack these layers to frame six layers of network architecture. We will dive into additional details underneath.

Fig.8 shows the design of our proposed CNN model. From the outset, we really want some pre-handling on the images like resizing it, normalizing the pixel values, and so forth. After the fundamental pre-handling, information is fit to be taken care of into the model.

Layer-1 comprises of the convolutional layer with ReLu (Rectified Linear Unit) initiation work which is the first convolutional layer of our CNN engineering. This layer gets the pre-processes image as the contribution of size n*n=32*32. The convolutional filter size (f*f) is 5*5, padding (p) is 0(around every one of the sides of the image), step (s) is 1, and the quantity of filter is 32. After this convolution activity, we get feature guides of size 32@28*28 where 32 is the quantity of feature maps which is equivalent to the quantity of filters used, and 28 comes from the recipe ((n+2p-f)/s) +1= ((32+2*0-5)/1) +1=28. Then the ReLu initiation is done in each feature map.

Layer-2 is the maximum pooling layer. This layer gets the contribution of size 32@28*28 from the past layer. The pooling size is 2*2; padding is 0 and stride is 2. After this maximum pooling activity, we get maps of feature of size 32@14*14. Max pooling is done in each feature map freely, so we get same number feature maps as the last layer, and 14 comes from a similar math formula ((n+2p-f)/s) +1. This layer has no function of activation.

Layer-3 is the second CNN layer with ReLu activation function. This layer gets the contribution of size 32@14*14 from the previous layer. The filter size is 5*5; padding is 0, the stride is 1, and the quantity of filters is 32. After this operation, we will be getting feature maps of size 32@10*10. Then, at that point, ReLu activation is going to be done in every mapping of feature.

As so on, in last layer, it is a completely connected and merged layer. This layer will process the class scores, bringing about a vector of size 8, where every one of the eight numbers relates to a class score, for example, among the eight classifications of Natural Images dataset. For definite results, we utilize the softmax as activation function.

Moreover, CNN transforms the first image layer by layer from the origin pixel values to the last class scores. Note that a few layers contain parameters, and others don't have it. Specifically, the convolution/completely connected layers perform transformation that are a component of the activations in the volume of input as well as of the parameters (the weights and biases of the neurons). Then again, the Relu/pooling layers will execute a proper function. We train the params in the convolutional/completely connected layers with stochastic inclination drop. By this interaction, we will set up the prepared model which will be utilized to perceive the image present in the test dataset or validation dataset. Accordingly, we can

characterize the image as our class or label - airplane, car, cat, dog, flower, fruit, motorbike and person.

e) Training, Validating & Testing the dataset

We have split the dataset into 70 – 30 percent. 70 is for training and 30 is for testing or we can say for validating data. After splitting the data. We have taken training data and based on that data we are creating some more artificial samples from it which is called Data Augmentation. In data augmentation techniques, we can create more samples from provided data by changing the angle of image to any direction randomly, we can also set random brightness, contrast and zoom in or zoom out. Like this we can make our model more accurate by creating and train the exist and augmented data together.

We have used libraries to validate the data in required format. For instance, CNN model needs same dimension images or samples to learn it. Because if it is not in same dimension then it will be very difficult for model to learn and cannot understand the data properly which cause the poor accuracy. We have resized the samples and make it ready for learn or training. We tried different paraments values and different epoch numbers to train it to get better accuracy.

Ultimately, we have tested the validate data by our model and we got around 92 percent accuracy which good to predict the image.

III. RESULT

As a result of our created CNN model, finally we are getting 92 percent accuracy. We have also tested the accuracy in different epochs, as we all know that, by increasing epoch number, model gets better. Sometimes it is not happened, but most of time it gets better accuracy after each epoch cycle. We have got the result after every 5 epochs as in figure 8 below:

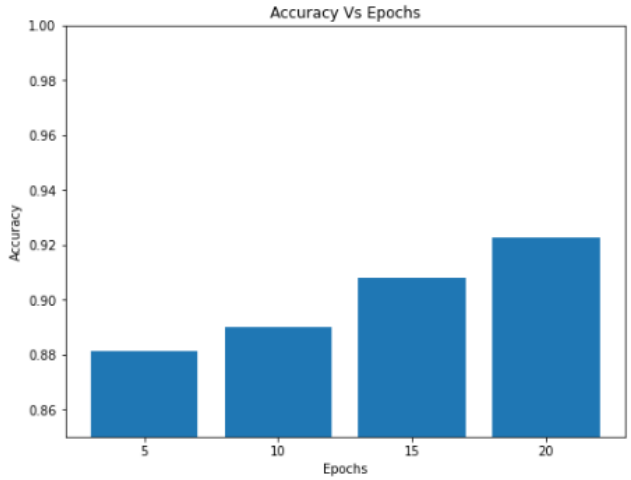

Fig. 8: Test dataset accuracy in various epoch cycle

We have plotted the graph in figure 9 for understanding in better way that how our model learns from training data and gets well trained by each epoch. It clearly shows that each epoch accuracy gets better than its last one and decreases the loss amount alongside. It started from around 75 percent accuracy. And after each cycle, accuracy increased majorly with next five epoch cycle, afterwards it increased gradually which shows that our model gets same information in learning that it has already learned in its previous epoch cycles.

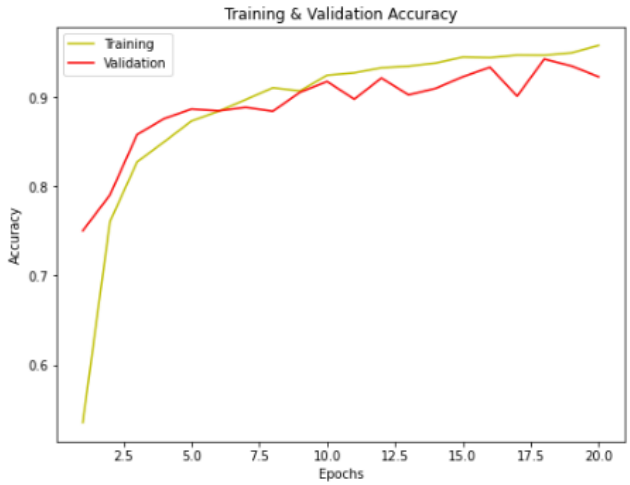

Fig. 9: Training Vs Validation accuracy at each epoch

The model gives 92 percent accuracy, so we need to check where our model got wrong in 8 percent. For that we have created confusion matrix in figure 8 which shows where our model gets confused and predicted wrong class or label.

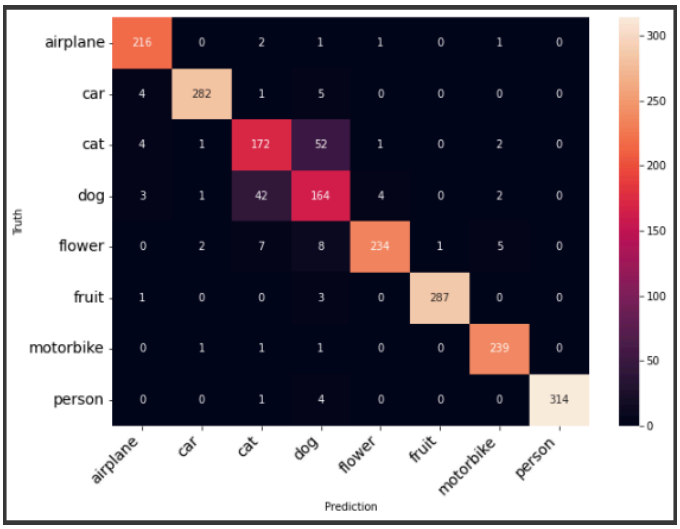

Fig. 10: Model confusion matrix

Above in figure 10, confusion matrix shows that, our model gets confused majorly to identify class dog and cat. Hence both are animal so it acceptable that our model will be getting wrong prediction to identify whether it is dog or cat. This is how we can get to know exactly where we are losing our accuracy and improve it by giving more data (samples) of dog and cat labels in our case.

IV. CRITICAL ANALYSIS OF RESULT

a) Analysis & Discussion

Among 30 percent test samples, our CNN model misclassifies only 8 percent images after 20 epochs which is good that we are having 92 percent accuracy recognition rate. These results are great for only 20 epochs and for such a simple algorithm with CPU training and less learning time (about 90 minute).

Although there are some images which are very difficult to recognize it, and out model is able to identify them correctly. For instance, we have many images with blurry pixels and small size image which is likely difficult to identify it.

Our validation accuracy is 92 percent which shows that our model has been trained well for prediction. Training sample size will affect the accuracy to increase as the number of samples increase. If there is more samples in training set, the smaller impact of training error and test error and finally the accuracy can be improved.

b) Achievement Process

We had started to create the model with default parameters with only 5 epochs and 2 convolutional layers where first layer had 16 filters and second had 32. With this configuration, we got 64 percent accuracy. Afterwards, we added more convolutional layers with 16, 32, 64 and epoch was 5 then we got 77 percent accuracy.

```
# Fetching total number of classes/labels
num_classes = len(class_names)

# Creating CNN model
model = Sequential([
  # Normalization layer
  layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),

  # Data Augmentation layer
  layers.RandomRotation(0.2),
  layers.RandomZoom(0.2),
  layers.RandomFlip('horizontal'),

  # Convolution & max pooling layer
  layers.Conv2D(8, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),

  # Dense or Flatten layer
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

Fig. 11: Snipped code of CNN model building

Finally as in figure 11, we set our model configuration to 4 convolutional filter layers with 8, 16, 32 and 64 alongside we put max pooling layer to focus on important features. Moreover, we also provided augmented samples and increased the number of epoch cycle from 5 to 20 and we got 92 percent accuracy which satisfied result. We can increase our model accuracy by increasing the epoch number and convolutional layers to it. However more layers and epochs CPU consumes more time to train it. It is like trial and error. It depends on dataset.

V. CONCLUSION

We have the image dataset called Natural Images provided by our professionals. In that we need to build a model or machine learning algorithm that can identify the image from group of various classes (category). For example, dog, cat or a person. So here, we have created model using CNN techniques.

Here we show a model which can identify or recognize and group the images. Later it very well may be stretched out for object acknowledgment, character acknowledgment, and real time detection system. Image classification is a significant stage to the immense field of AI and computer vision. As seen from the consequences of the analysis, CNN ends up being much better than different classifiers.

The outcomes can be made more exact by expanding the quantity of convolution layers and hidden neurons. We can perceive the item from hazy or we can say poor quality pictures by utilizing our model. Image recognition is a phenomenal model issue for finding out about neural network, and it gives an extraordinary method for growing further developed strategies of profound learning. Later on, we are intending to foster an ongoing image recognition framework.

REFERENCES

[1] Kuntal Kumar Pal, Sudeep K. S.(2016).” Preprocessing for Image Classification by Convolutional Neural Networks”, IEEE International Conference on Recent Trends in Electronics Information Communication Technology, May 20- 21, 2016, India.

[2] Hayder M. Albeahdili, Haider A. Alwzwazy, Naz E. Islam (2015).”Robust Convolutional Neural Networks for Image Recognition”, (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 6, No. 11, 2015.

[3] K Sumanth Reddy, Upasna Singh, Prakash K Uttam(2017).” Effect of Image Colourspace on Performance of Convolution Neural Networks”, 2017 2nd IEEE International Conference on Recent Trends in Electronics Information & Communication Technology (RTEICT), May 19-20, 2017, India.

[4] Marc’Aurelio Ranzato, Fu-Jie Huang, Y-Lan Boureau, Yann LeCun, ―Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition‖ CVPR, 2007.

[5] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov‖ Dropout: A Simple Way to Prevent Neural Networks from Overfitting‖ Journal of Machine Learning Research 15 (2014) 1929-1958.

[6] Fabien Lauer, Ching Y. Suen, and G´erard Bloch ―A trainable feature extractor for handwritten digit recognition‖ Journal Pattern Recognition 2007.

[7] M. Fischler and R. Elschlager, ―The representation and matching of pictorial structures, II IEEE Transactions on Computer, vol. 22, no. 1, 1973.

[8] Kaiming, He and Xiangyu, Zhang and Shaoqing, Ren and Jian Sun ―Spatial pyramid pooling in deep convolutional networks for visual recognition‖ European Conference on Computer Vision, 2014.

[9] Karen Simonyan and Andrew Zisserman ―VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE- SCALE IMAGE RECOGNITION‖ arXiv:1409.1556v5 [cs.CV] 23 Dec 2014.

[10] C. Couprie, C. Farabet, L. Najman, and Y. LeCun. Indoor semantic segmentation using depth information. Internatinal Conference on Learning Representation, 2013. 2.

[11] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng ―Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representation II.

MATLAB CERTIFICATES

MATLAB Onramp

Machine Learning Onramp

Deep Learning Onramp

Image Processing Onramp