Identification and Classification of Solar Panels in Satellite Imagery for Computer Vision

A.G Vinnakota, Y. Feng, R. Whang, S. Soriano Pérez, V. Venkatesh

Duke MIDS, Duke University, Durham, North Carolina, 27705

---

1. Abstract

In this paper, we discuss four different approaches to identify and classify solar panels in satellite images. We lay specific emphasis on two different feature extraction methods namely Histogram and Gradients (HOG) and Color Filter Feature Extraction (CFFE). We then use a K-Nearest Neighbour (KNN) Classifier in both cases to classify the images. These methods achieved an area under the curve (AUC) of 0.800 and 0.837 respectively. In our third approach, we develop an ensemble model combining both the earlier described feature extraction methods. The AUC in this case was 0.888. Finally we developed a Convolutional Neural Network (CNN) with an AUC of 0.983. We compare the different approaches and analyze where they performed well and where they can be improved.

---

2. Introduction

Due to the rising concern over the pollution from the current energy sources, solar power has been regarded as one of the most promising methods to provide energy at low pollution and low environmental footprint. As of the end of 2018, solar power accounted for around 1.6% of the total power consumed in the United States amounting to 64.2 GW. Furthermore, solar power has ranked consistently as the first or second fastest growing power source in the US since 2013[1]. Moreover, unlike nuclear, wind or hydrothermal energy solar energy can be easily harnessed from rooftops using photovoltaic cells. Solar power using photovoltaic solar panels is a quickly growing source of power across US households with increasing rates of adoption. Tracking the installation of solar panels across the nation to study the rates of adoption is often cumbersome and inaccurate - for such methods often involve self-reporting, monitoring sales of solar panels etc. A much more recent technique to track the harnessing of solar panels across the nation involves using machine learning models on satellite imagery. These models use a variety of techniques, both simple and intricate, to accurately identify the existence of solar panels in satellite images with high accuracy.

In order to obtain a thorough understanding of the current state of individual solar panel installation and usage both on a local and national scale, an automated method of identification is required for classifying the existence of solar panels based on pre-processed satellite images. In the notion of machine learning algorithms, the original model is built on the basis of an educated estimation and exploration of the dataset and trained multiple times on the training data with possible future stages of parameter tuning. The model performance will be evaluated based on its test data, which in this case are images unseen by the model beforehand. The finalized model will be able to provide classification on newly imported images in a scalable fashion. With the implementation of machine learning to solar panel image classification, the cost and time required to identify installed solar panels could be potentially reduced and will help the industry better provide renewable, clean and affordable alternative energy sources.

3. Background

Existing literature in this domain has attempted to estimate the number of solar panels in the U.S., map its geographic location and even go as far as estimate the size of solar panels installed. The Deep Solar Project is a successful initiative by researchers at Stanford University that used deep learning techniques, GPS coordinates and satellite imagery to construct a comprehensive high-fidelity solar deployment database for the contiguous U.S. According to their estimates as of December 2018, solar deployment density peaks at 1000 capita/mile^2, solar deployment increases with per capita income and asymptotes at ~$150K/year and has an inverse relation with the Gini index representing income inequality.[2] The Gini index is simply a measure of the income distributed across income percentiles. A high Gini index corresponds to greater levels of income inequality in the society approaching a Pareto distribution. Deep Solar has also developed a successful predictive model to predict future solar deployment based on census properties.

State of the art methods like those of Deep Solar use sophisticated deep learning techniques to achieve high rates of accuracy and low rates of incorrect classifications. These methods, however, are not easy to interpret and in some ways are analogous to a black box whose inner workings are a mystery. In contrast, our work attempts to highlight the importance of feature selection in model training and how effective feature selection coupled with rather simplistic classification methods such as a K-Nearest Neighbours (KNN) classifier and logistic regression can help achieve high levels of accuracy and low levels of inaccurate predictions. While the performance may not be identical to that of a Convolutional Neural Network (CNN), it will be comparable while simultaneously having the added advantage of being inferrable and interpretable. We hope our work will help the computer-vision community better understand the fundamentals of object detection and classification.

4. Data

The nature of this Kaggle challenge is image classification using 15,000 images of rooftops containing solar panels or not as training data and 508 images as public test data and 762 images as private test data. Some examples of the training images have been provided in Figure 1.

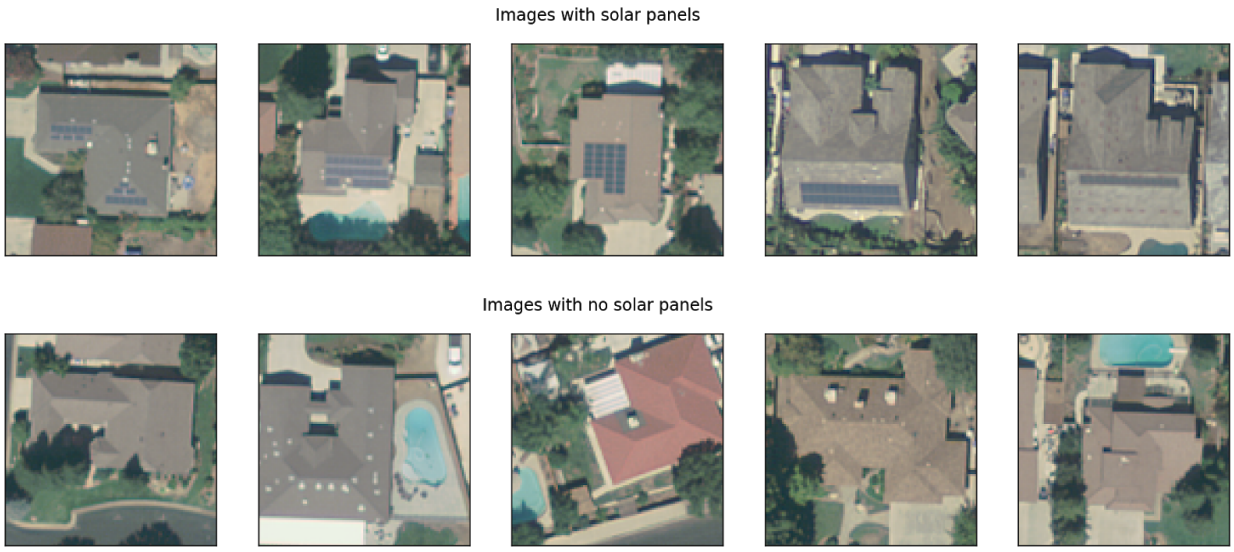

Figure 1. *Example of training data. Top: Images with solar panels (where y = 1). Bottom: Images with no solar panels (where y = 0).*

The participants are allowed to examine the performance of the model only on the public test date. The images provided contain pixels of 101*101 size in three panels of R, G, and B, each in the range of (0,255). There are a total of 30603 (101*101*3) features to choose from in the process of model training. With the number of samples being 15,000, the number of features significantly outweighs the number of samples. This is a situation called Curse of Dimensionality, meaning the number of predictors is much greater than the number of observations. Possible general solutions may include variable subset selection (filter, wrapper, ensemble models), and regularization and dimensionality reduction. We shall explore some of these possibilities later during model development.

Data imbalance is another factor that would potentially affect model performance. The training dataset appears to be imbalanced, with 505 images labeled as 1, meaning the image contains a solar panel and 995 images labeled as 0, meaning they do not contain solar panels (Figure 2). This imbalanced training data distribution may undermine performance on test data if it does not have a similar distribution.

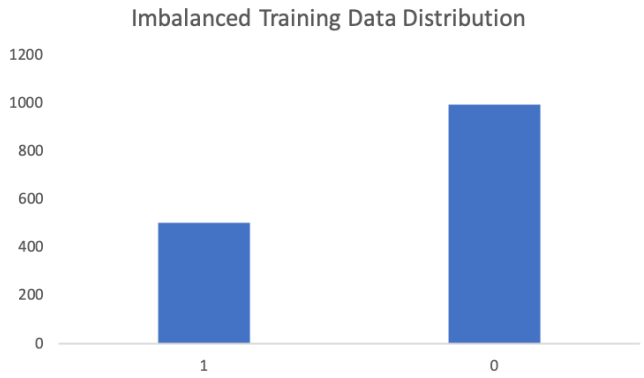

Figure 2. *Imbalanced Training Data Distribution. '1' refers to the images with solar panels and '0' refers to the images without solar panels.*

5. Methods

In order to process aerial imagery data into a form to ready it for machine learning algorithms, we consider histogram of oriented gradient (HOG) and Color Filtration Feature Extraction (CFFE). For training and testing multiple supervised learning techniques, we employ K Nearest Neighbors (KNN) and logistic regression methods. To compare and evaluate the performance of proposed schemes, we also investigate the convolutional neural network (CNN) method.

5.1. K Nearest Neighbors and Logistic Regression Approaches

We shall first explore three different models - CFFE coupled with a kNN classifier, HOG coupled with a kNN classifier, and an ensemble model that couples both HOG and CFFE with a logistic regression classifier. The following sections explain in detail the feature extraction and classification in all the three models.

5.1.1 Model 1: K Nearest Neighbors Model with Color Filter Feature Extraction

For the first model, we tried to considerably reduce the features on our training dataset by filtering out the colors in every image that did not lie inside a defined range of colors. Our hypothesis was that all solar panels on the satellite images had similar colors (a particular shade of gray), so filtering out other colors that were very different would help us identify the pictures with solar panels in them more easily. It was possible to define a lower and a higher color value for all three RGB channels and then create a mask of zeros and ones of the same 101x101 dimensions as the images in the training set. If a pixel in one of the images fell inside the range of lower and higher color values, the corresponding pixel in the mask would get a value of 1, if it did not it would get a value of 0. By multiplying each value of the mask times its corresponding

pixel on the image, we got a modified version of the images where every color outside the specified range became zero (black). We found an optimal value for the lower and higher filtering colors by iterating over different values and fitting a KNN model each time on the training data.

We picked the values with the best AUC performance, which are the following:

- Lower color value (RGB channels): 0, 0, 110
- Higher color value (RGB channels): 55, 87, 187

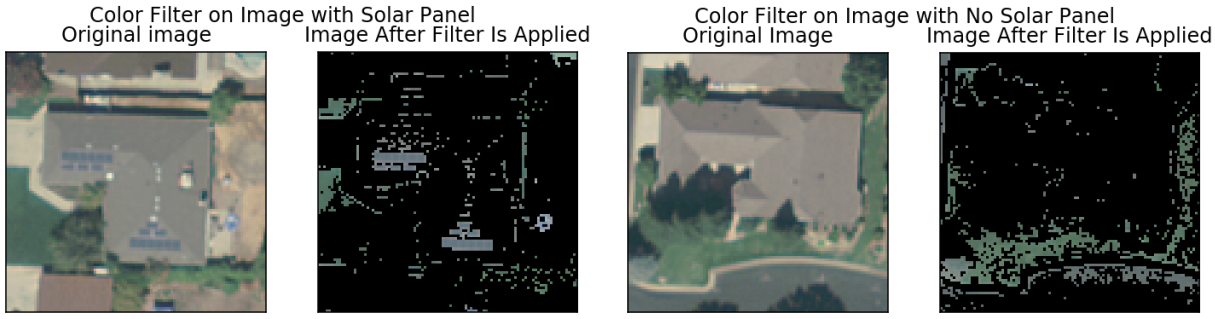

Figure 3. Left: *Image with solar panel before and after the color filtering.* Right: *Image with no solar panel before and after the color filtering.*

An example of the color filtering process can be observed in Figure 3. After the color filtering was done, we calculated the average value for the non black pixels in each image for each of the three RGB channels. Our hypothesis was that those images with a solar panel would have a significantly different average value for the red, green, and blue channels than those without solar panels, due to the presence of the specific shade of gray of the solar panels in the pictures. Consequently, the final features extracted with this process were three: the average value of red, green, and blue for each image.

With just three features per image now, we fitted a KNN algorithm with K = 7 on our new training data. We also tried to fit a logistic regression model but it gave worse AUC values than a KNN model. The KNN algorithm functions on the basis of assuming that similar things exist in close proximity. In other words, similar things are near to each other. KNN captures the idea of similarity (distance, proximity, or closeness) with some mathematics, calculating the distance between points on a graph. The advantages of KNN include its simplicity and easy implementation, wide range of usages and relatively low cost on parameter tuning. Since the input variables have been engineered to achieve dimensionality reduction, the disadvantage of fast-growing computational cost of KNN could be avoided[5]. Therefore, KNN classification is

applied in the second step following the features generated through the color filter process. All these steps are described in the following flowchart (Figure 4).

To test the model's performance on unseen data, we performed a K-fold cross validation assessment with K = 10. Due to the computational complexity of executing this same process on for the CNN model discussed in section 5.2, we also randomly split the training set into a smaller training set with 75% of the data and a validation set with the other 25%. The results are discussed in section 6.1.

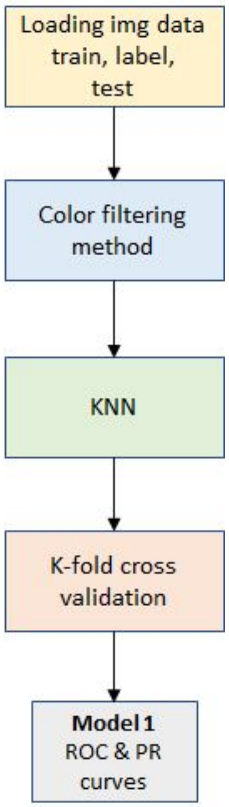

Figure 4. KNN Classifier with Color Filter Feature Extraction - Model 1

5.1.2 Model 2: K Nearest Neighbors Model with HOG

For the second model, the second approach of reducing dimensionality is the application of histogram of oriented gradients (HOG) to extract features in the imagery, then we fitted a K Nearest Neighbours (KNN) model on its output.

Feature engineering is one important factor of our model performance improvement as discussed above. The previous discussion of Curse of Dimensionality mentioned the excessive features could undermine the model, and the histogram of oriented gradients (HOG) method is

one potential solution since it reduces the information in the imagery while still allowing differentiating the images.

$$\text{Gradient}: \sqrt{G_x^2 + G_y^2} \qquad \text{Orientation}: \phi = arc\tan\left(G_y / G_x\right) \tag{1}$$

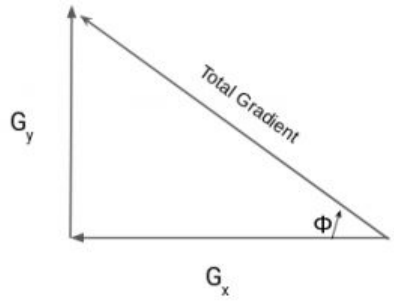

Figure 5. The calculation of HOG gradient and orientation

HOG method functions by focusing on the shape of the image. It is widely used in computer vision and object detection currently. It extracts the gradient and orientation of the imagery by breaking down the complete image into smaller regions and compute the gradients and orientations of the corresponding pixel values in these regions[3, 4, 7]. The basic math behind HOG is given above in Equation (1) and Figure 5 . The general steps of conducting HOG could be broken down into the following: Data Preprocessing, Gradient Calculation, Orientation Calculation, HOG calculation and completion of feature engineering[8].

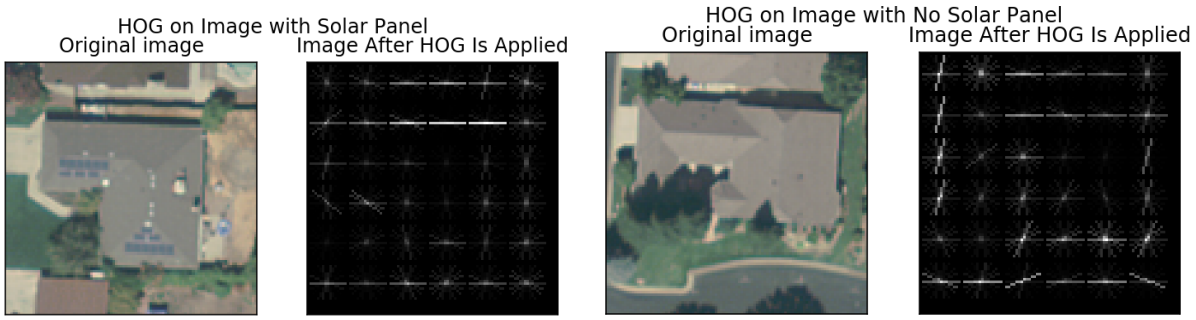

Figure 6. Left: *Image with solar panel and its corresponding map of gradients for each block*. Right: *Image with no solar panel and its corresponding map of gradients for each block.*

Some training images with HOG applied have been depicted above (Figure 6). Note how edges in the picture have been successfully detected. After completing HOG on the training dataset (Figure 7), the extracted key features were successfully in a reduced dimensionality and are ready to serve as input variables in the following stage of applying K Nearest Neighbours (KNN). This model could generate labels based on the closeness in proximity with relatively less training and tuning. The output engineered featured from HOG could also improve the performance of KNN, making it a great match for the combinations of these two techniques in our implementation.

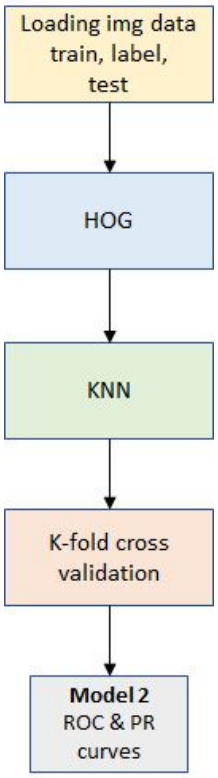

Figure 7. K Nearest Neighbors Model with HOG - Model 2

5.1.3 Model 3: Ensemble Model Using Logistic Regression

For the third model, we developed an ensemble model that used the outputs from Model 1 and Model 2 as inputs for a more accurate classifier. Our hypothesis was that the first model that used the color filter as a preprocessing and feature extraction method would correctly classify a certain type of images that the second model using HOG would not be able to do, and vice versa. By using the prediction scores calculated for the test/validation data from the first models as the features of our data, we could have more accurate predictions (since the features themselves would already be good predictors for the correct labels). The flow chart of the Ensemble Logistic Model is shown in Figure 8 below.

The main process of estimating coefficient in logistic model is through training a linear model to generate possibilities and sigmoid function to convert the result to ‘1’ or ‘0’ binary classifications based on related assumptions on data distribution. The advantages of this model include high efficiency, easy implementation and low requirement on computational resources. An important factor contributing to its improved performance is feature engineering, which has been conducted in the previous stages.

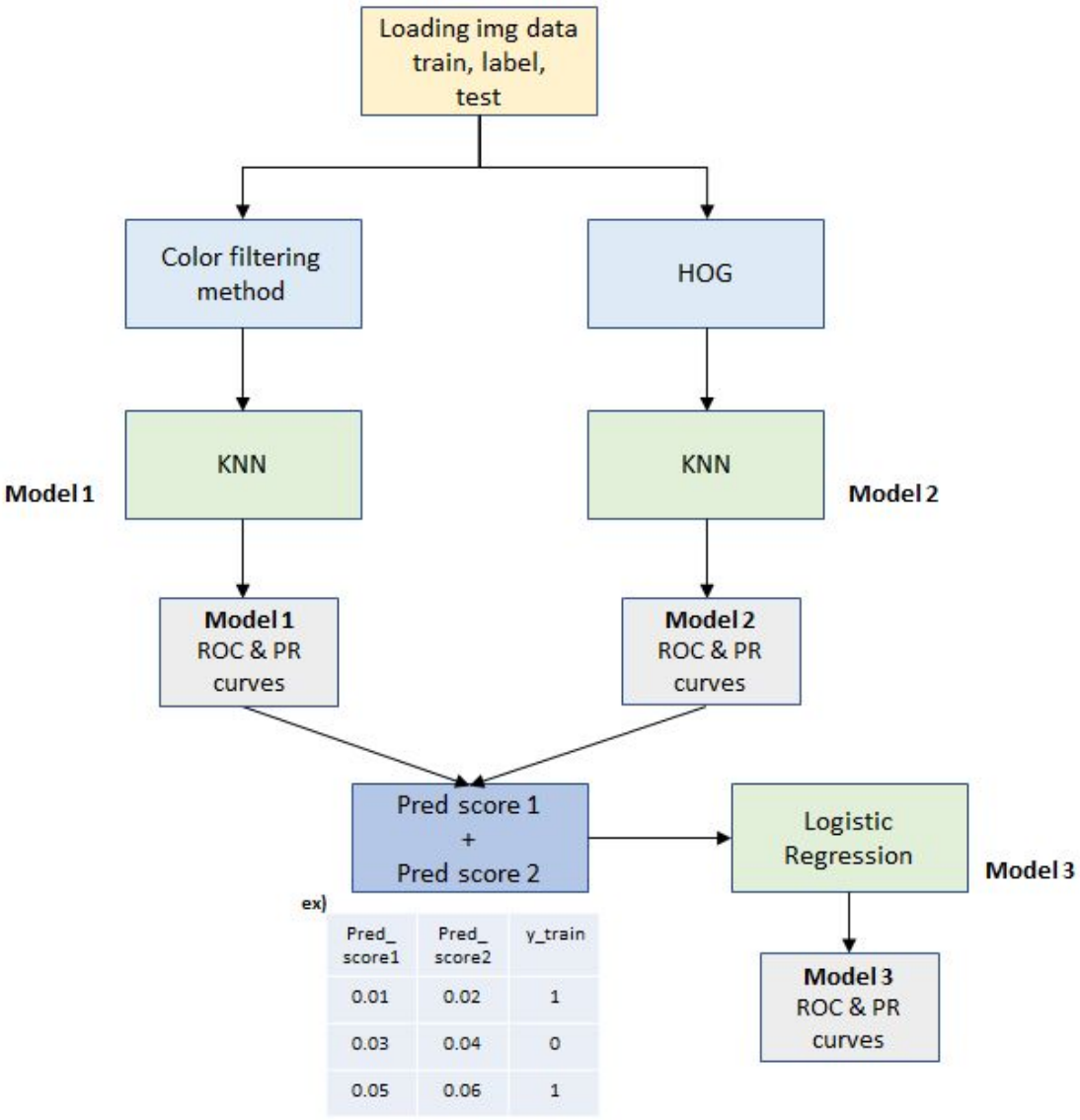

Figure 8. Ensemble Model Using Logistic Regression: Model - 3

5.2 Convolutional Neural Network Implementation[11]

The second type model that we used is a Convolutional Neural Network (CNN) to classify images. Image data by definition is high dimensional. In our specific case, each image is 101X 101 pixels amounting to roughly 10,000 different features. Simple Feed Forward Neural Networks (FFNN) will in general struggle with such high dimensional data. In contrast, CNNs do an excellent job of reducing the number of parameters while simultaneously doing an excellent job of classifying the images. In the following sections we shall explain in great detail the fundamental architecture and workings of our CNN.

5.2.1. RES-NET

We decided to derive inspiration from state-of-the-art neural network architectures out there and build our own neural network in hope to get a higher accuracy as compared to the baseline model.

Out of the many architectures, Res-Net appealed to us the most. It has 2 distinct advantages when compared to an architecture such as the VGG. One, it solves the vanishing gradient problem that occurs in deeper networks during back-propagation. Second, it makes the network unaffected by the varying size of the object of interest (solar panels) in different images by allowing for different receptive fields at any layer in the network. Both of these are achieved by the use of skip connections.

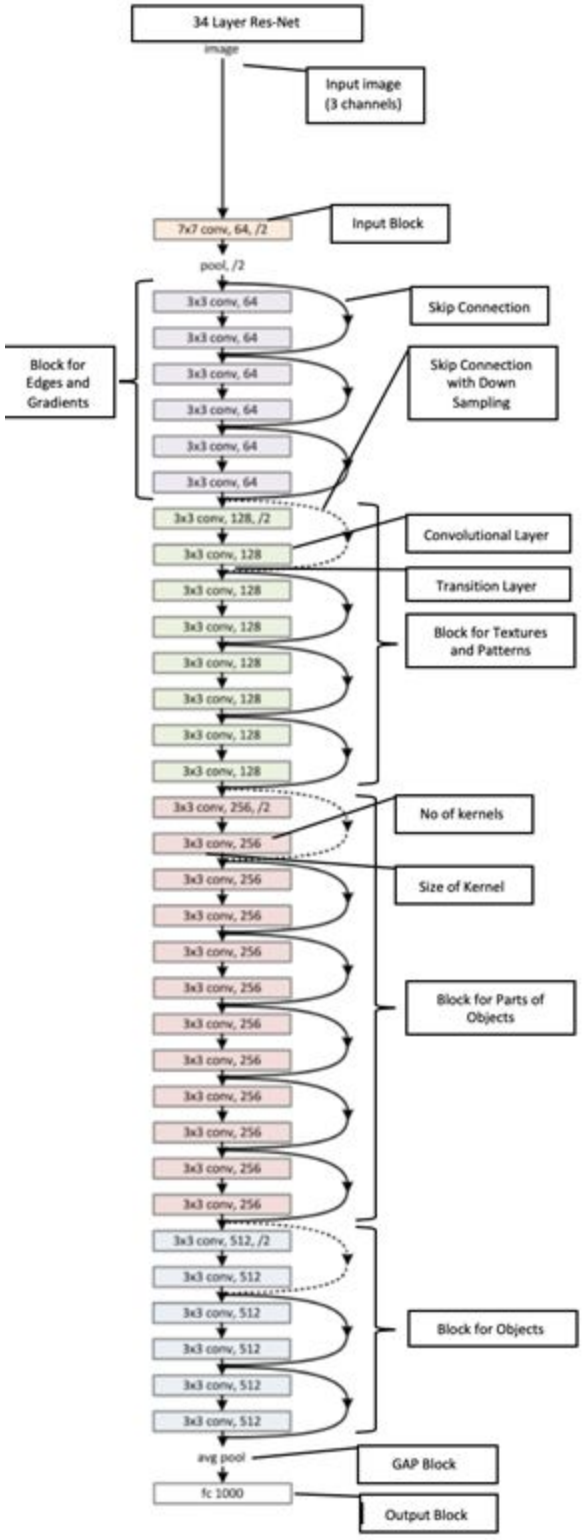

Figure 9. A 34-Layer Res-net used to classify images[11].

5.2.2. Various Components of CNN Architecture

1. Input Block
2. Block for Edges and Gradients
3. Block for Textures and Patterns
4. Block for Parts of Objects
5. Block for Objects
6. GAP Block
7. Output Block

5.2.2.1. Input Block

The primary goal of this block is to size down the original image into the appropriate size for which the network was designed. Top notch face recognition can be done on image sizes as small as 48*48 or 56*56 pixels. It shows that we do not need big images to achieve great classification. Therefore, we usually see big sized kernels being used to reduce the size of the image. Another important function is to convert the usual 3-channel (RGB) image into the required number of channels so as to pass it to the next convolutional layer.

5.2.2.2. Blocks (Edges, Gradients, Textures, Patterns, Parts of Objects and Objects)

It is interesting to note that the architecture of any neural network tries to imitate the actual human brains. Various experiments have been conducted to prove that the image that passes through the retina of the eye is inverted and projected at the back of the brain. First edges and gradients in the image are detected by the outermost layer of the brain, then textures and patterns, followed by parts of objects and objects. This happens as neurons transmit signals from the outermost parts of the brain to the innermost.

Each block usually consists of sequential convolutional layers and ends with a max-pooling layer. Max-pooling makes sure that only the features that really pop-out are passed through to the next block. Usually, it is advisable that at-least a receptive field of 5*5 pixels is got (can even go up to 11*11 pixels) before max-pooling is used, as it ensures an edge is well captured before max-pooling tries to reduce the size of the image. (A number of blocks can be

combinedly called as one block that detects edges and gradients as well, it is pretty subjective to the problem being solved at hand).

The following are the number of layers for each of the blocks in the above network (Figure 9):

1. Block for Edges and Gradients – 6 layers
2. Block for Textures and Patterns - 8 layers
3. Block for Parts of Objects – 12 layers
4. Block for Objects – 6 layers

The below image (Figure 10) shows visualized examples of feature maps (kernels) that give an idea of the kind of pattern that particular kernel tries to detect in that layer.

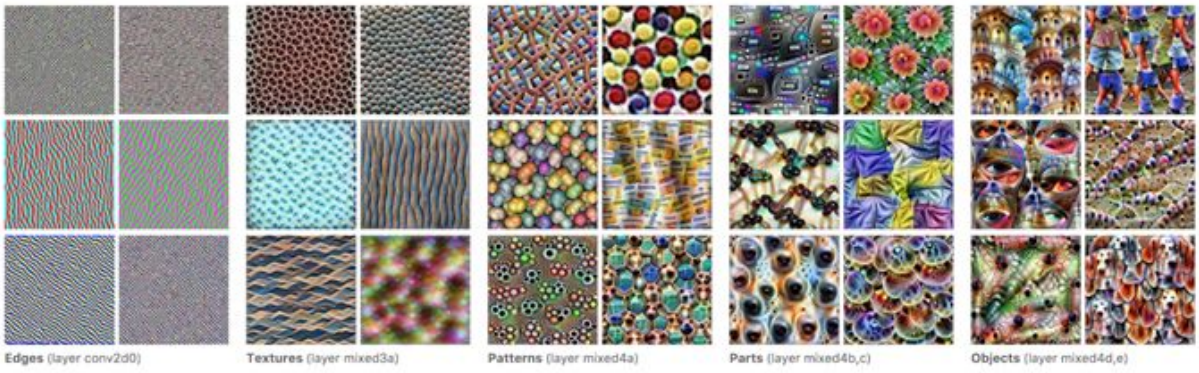

Figure 10. Feature maps for identifying edges, textures, patterns, parts and objects[11].

It is also interesting to note that the neuron complexity in our brain increases as we proceed inwards. This translates into using more kernels as we move from edges and gradients block to the objects block. It can also be understood in another way, as, more complex the part of the image that we need to detect more features (kernels) are required to capture it.

The following are the number of kernels used for each of the blocks in the above network (Figure 7):

1. Block for Edges and Gradients – 64 kernels

2. Block for Textures and Patterns - 128 kernels

3. Block for Parts of Objects – 256 kernels

4. Block for Objects – 512 kernels

It should be noted that the above network was designed for the image net dataset that has 1000 classes. Hence the number of kernels being used are so high. Usually, keeping in mind the computational efficiency the number of kernels used are a multiple of 8. It is also a common practice of doubling the number of kernels in case we want to increase the kernels within or across the blocks. Therefore, it follows a geometric progression such as, 16, 32, 64 and so on. However, in Res-Net, we do not increase the kernels in a particular block, but only across blocks. This is because of the addition of skip connections. More about this is discussed below. Each block consists of layers that fall into 2 broad types. The convolutional layer and the transition layer.

1. **Convolutional Layer**: The convolutional layer takes in input images, runs kernels on them to extract features. The input channels, required output channels, kernel size (usually 3*3*a wherein 'a' is the number of input channels of an image), number of kernels (equal to required output channels), stride, padding, bias are some of the hyper-parameters that are defined. Also, regularization techniques like batch normalization and dropout can be used. An activation function needs to be defined for each layer.

2. **Transition Layer**: The transition layer is responsible for incorporating the data from the skip connections and the convolutional layer. The way this is done is via simple element-wise addition of each image coming from both sides. For ensuring the possibility of element-wise addition, padding is used in the convolutional layers to make sure the 2D image size does not shrink, while the same number of kernels are maintained to make sure the 3rd dimension of the image does not change.

The above is applicable when the transition layer is present within a block (skip connections have straight lines). However, while moving from one block to another, we usually reduce the size of the image by half (this can successfully be done without losing much information from the image). Hence, when it comes to a transition layer that exists between 2 blocks, the data from the convolutional layer is put through either max-pooling or a 1*1 kernel to reduce its size. The data coming from the skip connection is down sampled (can be done by

using a 1*1 kernel with a stride of 2) to match the other data and make element-wise addition possible.

A note on Skip Connections:

Skip connections are like freeways in neural networks that allows free passage of info forward and backward. This is the cornerstone for the Res-net architecture. These connections offer 2 major advantages:

1. **Solves the vanishing gradient problem:** The vanishing gradient problem is caused by back-propagation during parameter tuning. Back propagation tries to alter the weights assigned to the parameters in order to minimize loss. It does this by taking the gradient of the parameter with respect to the loss and tries to take a step towards minimizing loss. As we move to the preceding layers, the chain rule comes into play, i.e., the product of the local gradient and the gradient flowing from the head. Now, if the local gradient is anything less than one, this product keeps on decreasing and vanishing as we move across more layers. This causes a lot of inefficiency in the training process which is handled by the skip connection as it introduces an identify function. The gradient of the identify function is 1, which prevents the gradient from disappearing at all.

2. **The network learns a combination of low level and high-level features:** In order to better appreciate this, one must understand the concept of a receptive field. The receptive field of a particular pixel is defined as the number of pixels it has been exposed to of the original image. The following figure (Figure 11) will better help you understand the concept.

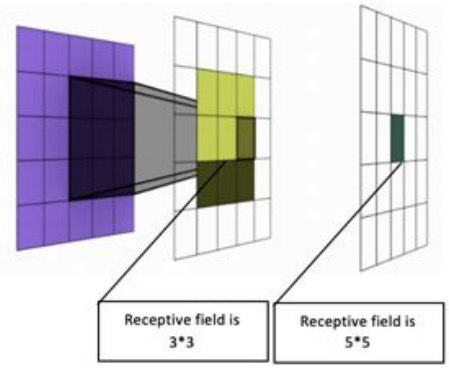

Figure 11. Figure explaining the concept of receptive fields[11].

Now, the skip connection enables element-wise addition of the same image, one that has come from the convolution block and one that has come directly via the skip connection without any convolutional layers in between. Pixels with lower receptive fields are being combined with

pixels with a higher receptive field. This exposes the network to a combination of low-level and high-level features which it can learn effectively. In other teams, the network becomes more resilient with respect to different sizes of the object of interest in any given image.

5.2.2.3. GAP Block

Once we are happy with the layers and feel that we have built a sufficient model, we move to this block. Earlier, we used to use a fully connected layer here, i.e., stretch the image into a string and input that into a function that gives out a confidence score for a particular class. However, there are various disadvantages for this. The location information of the pixel gets lost, i.e., the method is not rotational or translational invariant. Hence, the global average pooling is used nowadays to achieve the same. It is nothing but an operation of averaging the pixel values across all the channels of an image to output just on channel. This channel will be given as input to the output block.

5.2.2.4. Output Block

In the output block, the single channel image is now stretched into a string and two 1*1 kernels are run on it to give 2 outputs, one meant for each class. This output is used as input for a function that would give out the confidence scores for each of those classes.

5.2.3. Architecture of our CNN

Based on our exploration of what the RES-NET can achieve and how it accomplishes the same, we went ahead and designed our network based on it. The overall architecture can be seen below (Figure 12).

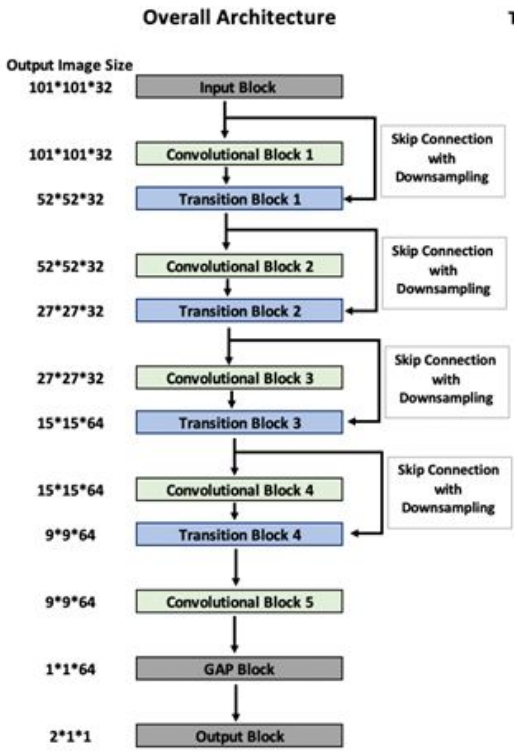

Figure 12. The architecture of our CNN[11].

We understood that the architecture we studied was meant for ImageNet dataset for 1000 classes. As the number of classes for us was just 2 and the dataset small, we decided to keep the model modest. We used 5 convolutional blocks, wherein each consists of 2 layers of convolutions followed by one layer of down sampling. We only used skip connections across blocks and none in-between the blocks. We started out with just 32 kernels for the first 3 blocks, post which we increased them to 64.

The following (Figure 13) are the structures of the first convolution block and transition block that will help understand the network better. All the blocks are of similar structure. The only thing that changes is the size of the input images.

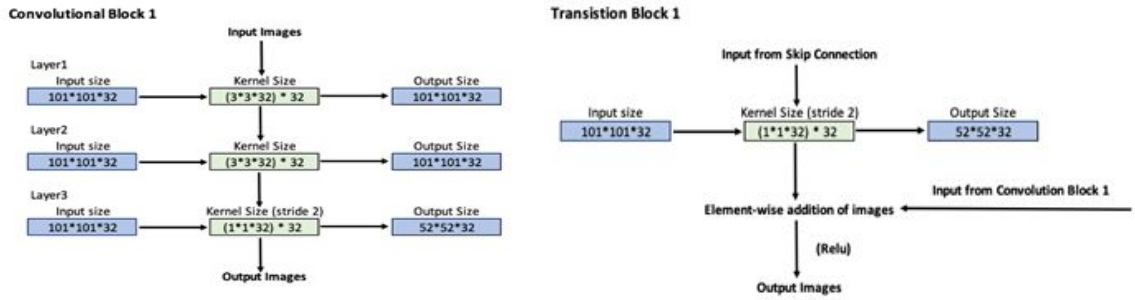

Figure 13. The architecture of a convolutional block and transitional block[11].

5.2.4. Design Considerations

We have made various choices in the design based on our understanding

5.2.4.1. 3x3 Kernel

A kernel (Figure 14) is nothing but a set of values (parameters) that are responsible for extracting a particular feature of the image. The below image to the left, shows a particular instance of the convolution process. The one to the right shows how the calculations happen, between the kernel and the image.

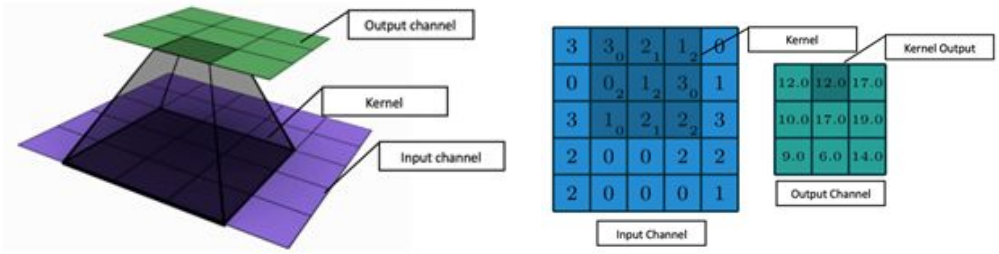

Figure 14. Justifying our choice of a 3X3 kernel[11].

We have chosen the 3*3 kernel for the purpose of convolutions. There are 3 reasons behind this:

1. The present GPUs are best optimized for 3*3 kernels. It has pretty much become the industry standard.

2. Using the smallest possible kernel would give you the highest flexibility in your network. However, we need the size of kernel to be an odd number. This is because we need a center so that gradient can be calculated. The smallest kernel that satisfies this is a 3*3 kernel.

3. They need lesser parameters, i.e., they are very efficient. Comparing Figure 11 and Figure 14, we can see that in order to convolve on a 5*5 image to get a single pixel, we need 2 3*3

kernels, i.e., 18 parameters. However, in order to do the same with a 5*5 we would need 25 parameters.

5.2.4.2. 1x1 Kernel

A 1*1 kernel is a special kernel. It does not increase the receptive field of the pixels as such. Hence, not useful for convolution. However, they are very useful when we need to alter the channels of an input. From the below image, you can see that it can assign different weights to different channels of the same pixel and give an output. This can be done on the entire image by any number of different kernels. Therefore, it's a pretty good way to after the channels. We have specifically used these kernels in order to down sample in both the convolutional and the transitional blocks. That is only possible when we use a stride bigger than 1 (covered in the next section).

5.2.4.3 Convolutions (Stride)

A good analogy for a convolution is a duster erasing a whiteboard entirely, wherein, the duster can be compared to the kernel and the whiteboard the image. It can be understood by looking at the below set of images (Figure 15).

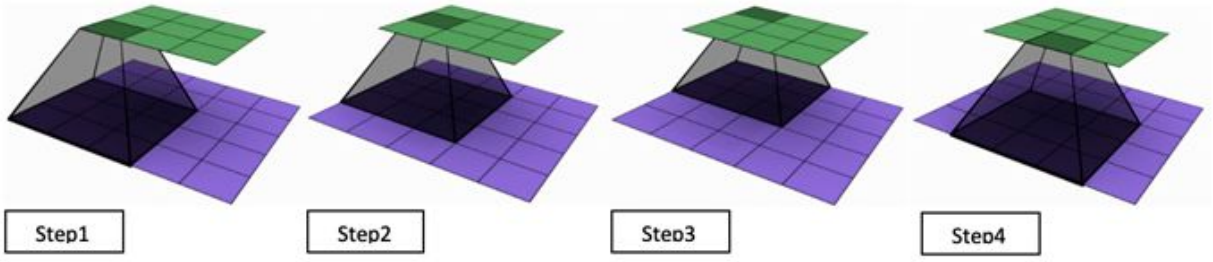

Figure 15. Filter convolving a patch with stride equalling 1[11].

The above convolution is happening at a stride 1 (Figure 14), i.e., it is moving by a pixel each time. If the stride is increased to 2, the convolution process jumps from step 1 to step 3 and so on. As such, we have used a stride of 2 with 1*1 kernel for down sampling in both the transitional as well as the convolutional blocks.

5.2.4.4 Channels

This basically refers to the 3rd dimension of the image matrix. Gray-scaled images have a single channel. Digital images usually have 3 channels (RGB). The same image in a newspaper would have 4 channels (CMYK). This basically means that a single image can be represented in

any number of channels, without altering it. This is the basis on which the pixel values in different channels get combined vis different weights coming from the kernel.

The below image (Figure 16) is a good representation of the mathematics behind a 3*3*3 kernel, giving an output channel.

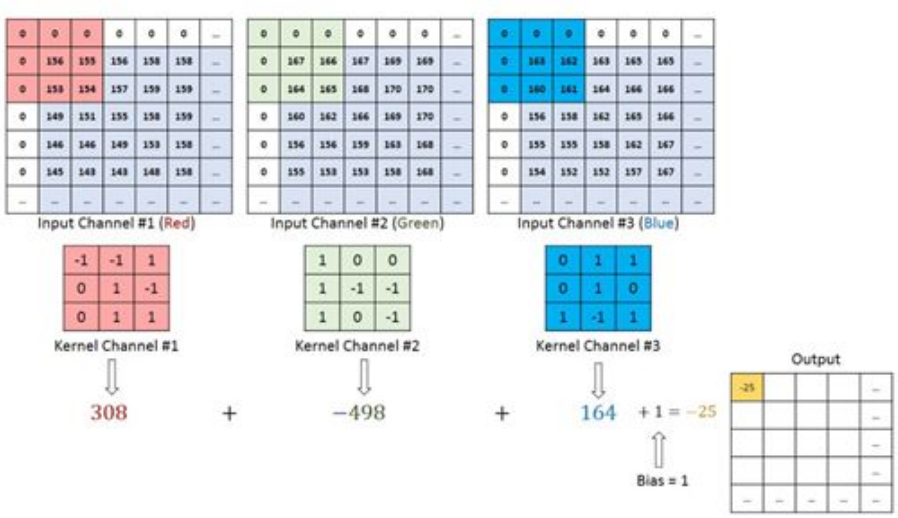

Figure 16. A kernel with 3 channels giving an output channel[11].

We can run as many kernels as we want, on the same image to get different outputs. These kernels can be referred to as a feature extractor, extracting different features / outputs from the same image. The following image is a good depiction of the same (Figure 17)..

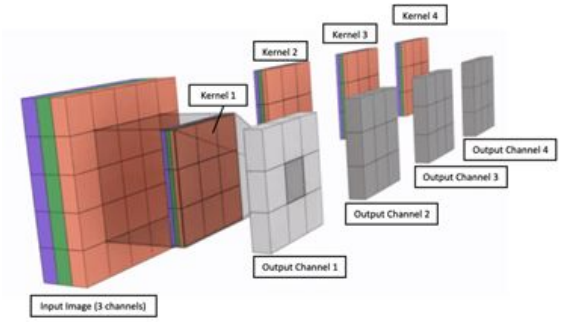

Figure 17. Feature extractor extracting different features from the same image[11].

The output channels are now stacked to form an image, with 4 channels (Figure 18). Remember, we can represent the same image in any number of channels we want to.

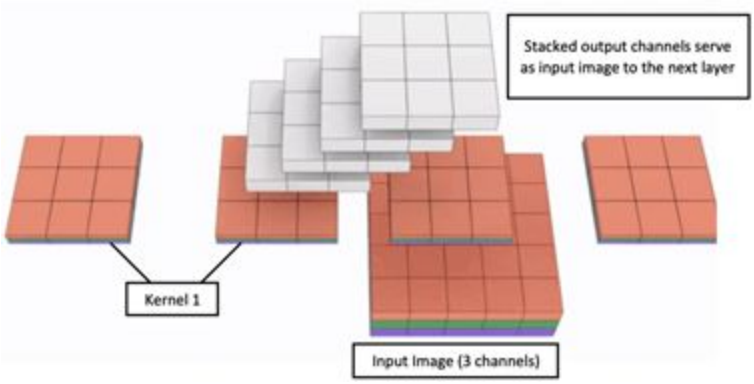

Figure 18. Output image with four channels[11].

It is good to note that the 3rd dimension of the kernel should be equal to the input image and the number of kernels used will determine the number of channels of the output image.

5.2.4.5 Padding

One would have noticed by now, that the size of an image decreases after every convolution. However, for transition blocks to carry out element-wise addition, it is imperative that the image size remains the same. Hence, we used padding in every convolutional layer in our network.

The below image (Figure 19) is a good example that depicts padding very clearly. It is nothing but adding an additional row or column of 0 valued pixels.

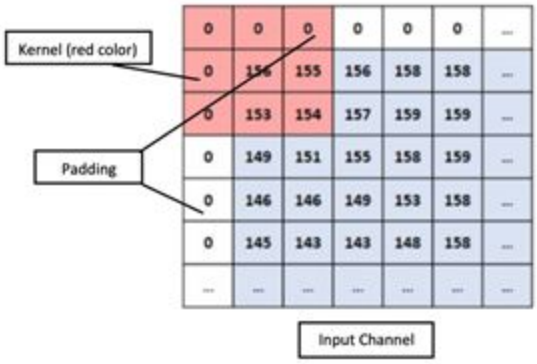

Figure 19. Input image with 1-layer of padding around[11].

5.2.4.6 Bias

The below image (Figure 20) is a snip of the image shown in the channel section wherein the computation was discussed. Here, we may choose to add or not add bias.

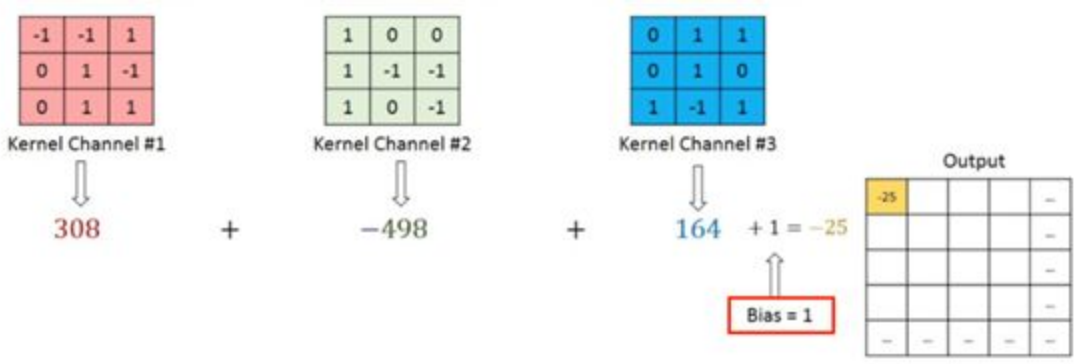

Figure 20. Depicting a filter with a bias. We may or may not choose to have a bias[11].

A parallel to adding bias in the step is like choosing to have an intercept in a linear regression function. If you choose to add bias, there will be more flexibility for the model as it is not bound to. We chose not to add bias as we have used batch normalization (regularization technique) in every layer which nullifies the bias anyways.

5.2.4.7 Dropout

Dropout (Figure 21) and Batch normalization are 2 regularization techniques we've used to prevent overfitting. While using dropout, we need to give a fraction value that indicates the fraction of kernels that needs to be ignored at random during the training phase. Each time a set of images pass through a different set of neurons are ignored at random. In our process, we first tried a drop out value of 5% , and then ended up using 10% for our final model.

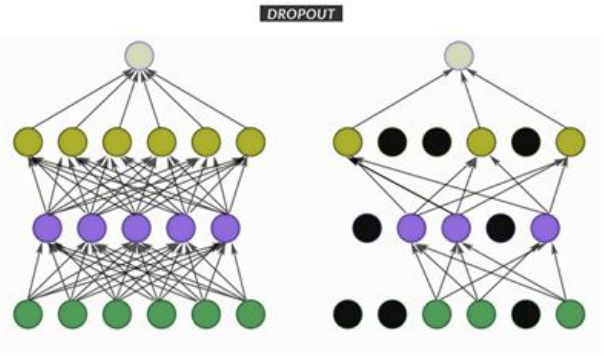

Figure 21. Dropout[11]

5.2.4.8. Batch Normalization

Batch Normalisation is the process of normalising the intensity of the pixels in the input training data. Under the hood, batch normalization takes the maximum and minimum intensity pixel in a patch. It will then find a factor that scales the minimum intensity pixel to black and the maximum intensity pixel to white. This in a way has the effect masking weaker features by setting them to 0 intensity and enhancing stronger features by setting them to 255 - the maximum intensity. Furthermore, batch normalisation has the added effect of accelerating the training speed of deep neural networks by reducing the number of epochs required to train.

5.2.4.9. Activation Function

Considering the goal of this analysis is to provide binary predictions for solar panel identification. Activation function could be highly useful since it mainly functions as an output signal to transfer its inputs to desired results that matches the exceptions for answering the questions. Combining Neural Networks, which was regarded as a universal function approximators, its results could be highly useful under various circumstances. One of its main applications include transforming linear inputs from Neural Networks to non-linear outputs more appropriate for classifications problems.

The advantages of applying activation functions include its easy implementation in terms of computational power, and it is faster to converge. These advantages could further aid in generating the accurate classification prediction results, which is the reason why activation function will be applied in our model.

5.2.4.10. SoftMax Function

In order for generating the likelihood of the variables being in a certain class. With the additional layer of Neural Networks, the complexity of input data could be potentially increased, making it harder to interpret. This is where SoftMax becomes useful by producing confidence scores that could later be directly used to make predictions, which is of high importance in our further analysis.

6. Results

In this section we will compare the results obtained for all the models described in section 5. First, we will present and compare the overall results comparing all models, including a random guessing model that predicts zeros and ones equally, regardless of the input features. Then, in section 6.1 we explain in greater detail the results of models 1, 2, and 3, looking at

examples where each method worked well and where it failed. Finally, we will discuss the results of the CNN model in section 6.2.

To evaluate the models' performance we performed k-fold cross validation with k = 10 on the training data, to estimate each model's performance on unseen data and used their AUCs to compare them. We encountered a computational problem when trying to perform k-fold cross validation for the CNN model, as it took a lot of time and computational power to execute and was therefore unfeasible. Figure 22. shows the ROC curves for the non-CNN models' cross-validation.

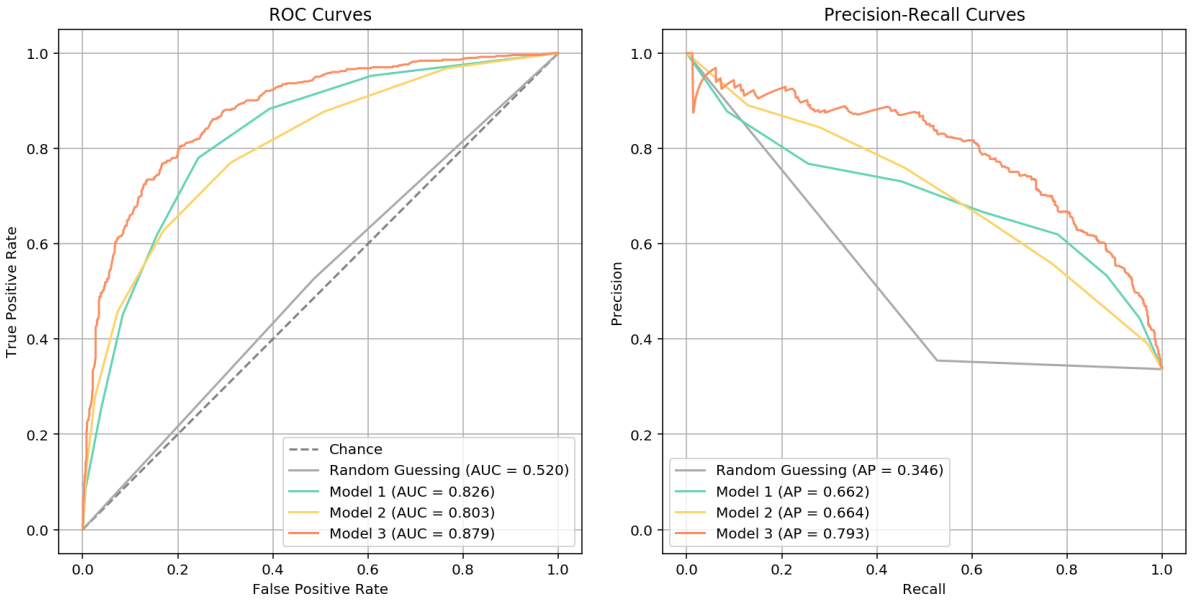

Figure.22. ROC and Precision-Recall Curves for cross validated performance.

		Predicted Labels	
		0	1
Actual Labels	0	125	124
	1	63	63

Figure 23. Confusion matrix for Random Guessing

The random guessing model predicts zeros and ones with equal chances, regardless of the input features. Its confusion matrix is shown in Figure 8. It has an accuracy of 50%, a precision of 34%, a recall or true positive rate of 50% and a false positive rate of 50%.

For the models explained in section 5.1, their performances are evaluated in terms of ROC and Precision-Recall curves. In ROC curves, Model 1 which is using color filtering with KNN is better than Model 2 using HOG and KNN, and ensemble model, Model 3 shows the best performance. Model 3's Precision-Recall curve shows that it performs better than any other models, and depends on threshold, performances of Model 1 and Model 2 keep changing. Random Guessing is the worst in any case.

Because, the CNN could not be cross-validated the performance of the CNN cannot be directly compared with that of the other cross-validated models. Therefore we performed a train/validation split on the data and retrained all the other models to facilitate comparisons with CNN. The ROC and PR curves for these models have been included below in Figure 24. It can be observed that the AUC values for the non-CNN models are consistent with the cross-validated results shown in Figure 23, while the CNN model outperforms all other three models.

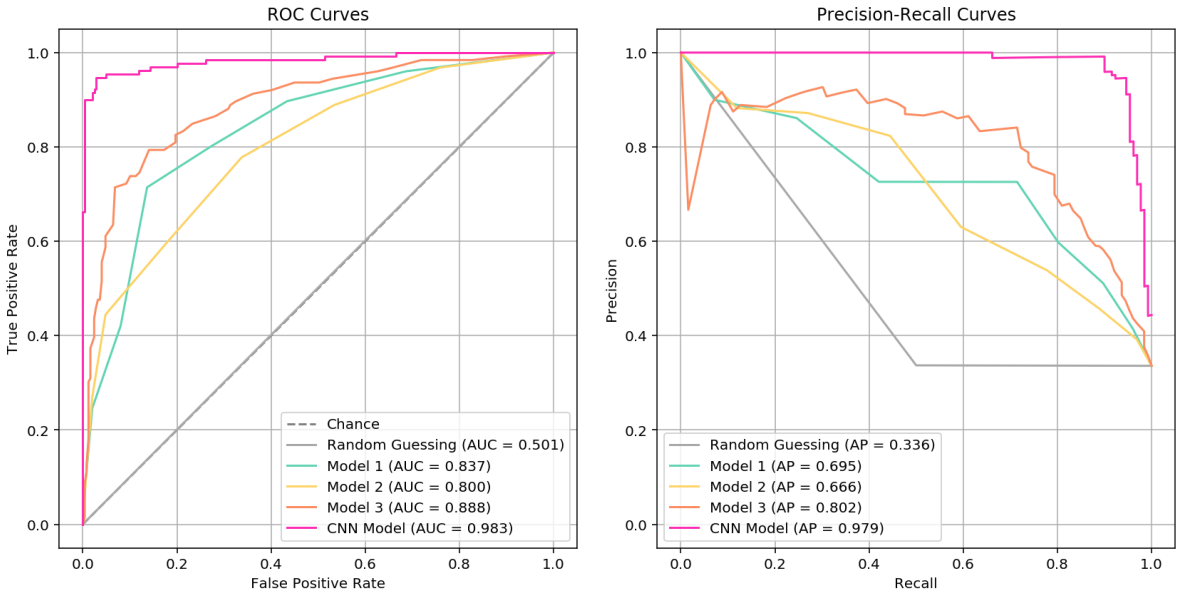

Figure.24. ROC and Precision-Recall Curves for validation set performance.

Models 1 to 3 utilized more basic approaches for image preprocessing and feature extraction, while using methods like KNN and logistic regression as classifiers. Although the results achieved with these non-CNN approaches are quite decent and much better than random guessing, they are not as good as the convolutional neural network approach. Model 3, which combines the results from the previous two models, achieves an AUC value of 0.888 on the validation set. If we use this model as the baseline, the AUC of 0.983 achieved by the CNN model turns out to be much better at correctly classifying the images and detecting solar panels. Similarly, the average precision value of 0.979 achieved by the CNN model is considerably

superior to the 0.802 value achieved by model 3. The CNN model's complexity seems to pay off by providing much more accurate results, that are discussed in greater detail in section 6.2.

6.1 K Nearest Neighbors and Logistic Regression Approaches

6.1.1 Model 1: K Nearest Neighbors Model with Color Filter Feature Extraction

		Predicted Labels	
		0	1
Actual Labels	0	215	34
	1	36	90

Figure 25. Confusion matrix for KNN with Color Filter

As for the K Nearest Neighbors Model with color filtering, the model is generally performing well with the optimal cutoff value that maximizes accuracy being 0.429. The accuracy of the result is 81.3%.And the True Positive Rate is 71.4% while the False Positive Rate is 13.7%, the modeling is doing a better job in classifying images that do not contain solar panels. It is evident that this model also performs much better than the random guessing model reviewed in section 6.

Model 1 uses color range filtering to extract features from the original images. It does well at detecting panels where the images show the panels inside the specific color range, and the surroundings do not have a lot of similar colors (there are more green and blue tones). However, if fails to detect images where there are similar tones in the surroundings (where it detects false positives) or the panels are not the same color as most other panels (it classifies the images as false negatives) (Figure 26). However, it does good at detecting images where the panels are clear.

Figure 26. True Positives, True Negatives, False Positive and False Negatives from Model 1.

6.1.2 Model 2: K Nearest Neighbors Model with HOG

		Predicted Labels	
		0	1
Actual Labels	0	205	44
	1	51	75

Figure 27. Confusion matrix for KNN with HOG

In the K Nearest Neighbors Model with HOG, the model is performing better with HOG than the previous model with optimal cutoff being 0.429 as well. In terms of accuracy, the result is 74.7%. And the True Positive Rate is 59.5% while the False Positive Rate is 17.7%. Similarly, the modeling is performing better when classifying images that do not contain solar panels. This model also improves on the results of the random guessing model explained in section 6.

Model 2 uses HOG to detect gradients and edges in the images. It does well at detecting panels where there is a clear contrast between the panels and the roof where they are located (true positives) (Figure 28), and if no such contrast is found it detects true negatives correctly. However, if there are shadows in the images that create similar contrasts with the surroundings it will detect false positives for images with no solar panel and it will miss panels when the orientation of the buildings in the pictures is not aligned with the edges, or the contrast is not strong enough.

Figure 28. True Positives, True Negatives, False Positive and False Negatives from Model 2.

6.1.3 Model 3: Ensemble Model Using Logistic Regression

		Predicted Labels	
		0	1
Actual Labels	0	206	43
	1	26	100

Figure 29. Confusion matrix for Ensemble model with Logistic

The Ensemble Model Using Logistic Regression has an optimal accuracy of 81.6% with a cutoff threshold of 0.378, a true positive rate of 79.4% and a false positive rate of 17.3%. The model is showing improvement in terms of accuracy. This model is an improvement over the random guessing model reviewed before.

Model 3 takes advantage of both the HOG and the color filter method. It detects panels well if the contrast between colors is strong (Figure 30), if the panels fall into the color range defined in Model 1 and the surroundings are of different colors. It still returns false positives when shadows or other objects look similar to solar panels and false negatives when the panels are too small and do not contrast much with the roofs and the surroundings.

Figure 30. True Positives, True Negatives, False Positive and False Negatives from Model 3.

6.2 Convolutional Neural Network Implementation

Once the model architecture was finalized, we had to train our model. We decided on an image batch size of 32. We decided to start with 150 epochs and then see if we needed to do more. We first ran the model with a dropout of 10%. Then we changed the drop out to 5% that raised the model accuracy that can be seen below.

		Predicted Labels	
		0	1
Actual Labels	0	233	12
	1	7	123

Figure 31. Confusion matrix for CNN (on validation set)

The CNN model has an optimal accuracy of 94.9% with a cutoff threshold of 0.164, a true positive rate of 94.6% and a false positive rate of 0.05%. By all metrics, this is the best performing model when compared to the models discussed previously.

This model went on to perform above 99% on the public leader board and around 98% on the private leaderboard.

We trained our model on 75% of the training data and validated it on the remaining 25%. The best model turned out to be the one we got at Epoch 131, which we ended up using as the final model on the validation set.

We did not see much value in further training for more epochs, evident from the graphs in the next page (Figure 34).

Here is the textual model summary of the neural network (Figure 32):

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 101, 101]	864
BatchNorm2d-2	[-1, 32, 101, 101]	64
ReLU-3	[-1, 32, 101, 101]	0
Dropout-4	[-1, 32, 101, 101]	0
Conv2d-5	[-1, 32, 101, 101]	9,216
BatchNorm2d-6	[-1, 32, 101, 101]	64
ReLU-7	[-1, 32, 101, 101]	0
Dropout-8	[-1, 32, 101, 101]	0
Conv2d-9	[-1, 32, 101, 101]	9,216
BatchNorm2d-10	[-1, 32, 101, 101]	64
ReLU-11	[-1, 32, 101, 101]	0
Dropout-12	[-1, 32, 101, 101]	0
Conv2d-13	[-1, 32, 52, 52]	1,024
BatchNorm2d-14	[-1, 32, 52, 52]	64
Dropout-15	[-1, 32, 52, 52]	0
Conv2d-16	[-1, 32, 52, 52]	1,024
BatchNorm2d-17	[-1, 32, 52, 52]	64
Dropout-18	[-1, 32, 52, 52]	0
Conv2d-19	[-1, 32, 52, 52]	9,216
BatchNorm2d-20	[-1, 32, 52, 52]	64
ReLU-21	[-1, 32, 52, 52]	0
Dropout-22	[-1, 32, 52, 52]	0
Conv2d-23	[-1, 32, 52, 52]	9,216
BatchNorm2d-24	[-1, 32, 52, 52]	64
ReLU-25	[-1, 32, 52, 52]	0
Dropout-26	[-1, 32, 52, 52]	0
Conv2d-27	[-1, 32, 27, 27]	1,024
BatchNorm2d-28	[-1, 32, 27, 27]	64
Dropout-29	[-1, 32, 27, 27]	0
Conv2d-30	[-1, 32, 27, 27]	1,024
BatchNorm2d-31	[-1, 32, 27, 27]	64
Dropout-32	[-1, 32, 27, 27]	0
Conv2d-33	[-1, 32, 27, 27]	9,216
BatchNorm2d-34	[-1, 32, 27, 27]	64
ReLU-35	[-1, 32, 27, 27]	0
Dropout-36	[-1, 32, 27, 27]	0
Conv2d-37	[-1, 32, 27, 27]	9,216
BatchNorm2d-38	[-1, 32, 27, 27]	64
ReLU-39	[-1, 32, 27, 27]	0
Dropout-40	[-1, 32, 27, 27]	0
Conv2d-41	[-1, 64, 15, 15]	2,048
BatchNorm2d-42	[-1, 64, 15, 15]	128
Dropout-43	[-1, 64, 15, 15]	0
Conv2d-44	[-1, 64, 15, 15]	2,048
BatchNorm2d-45	[-1, 64, 15, 15]	128
Dropout-46	[-1, 64, 15, 15]	0
Conv2d-47	[-1, 64, 15, 15]	36,864
BatchNorm2d-48	[-1, 64, 15, 15]	128
ReLU-49	[-1, 64, 15, 15]	0
Dropout-50	[-1, 64, 15, 15]	0
Conv2d-51	[-1, 64, 15, 15]	36,864
BatchNorm2d-52	[-1, 64, 15, 15]	128
ReLU-53	[-1, 64, 15, 15]	0
Dropout-54	[-1, 64, 15, 15]	0
Conv2d-55	[-1, 64, 9, 9]	4,096
BatchNorm2d-56	[-1, 64, 9, 9]	128
Dropout-57	[-1, 64, 9, 9]	0
Conv2d-58	[-1, 64, 9, 9]	4,096
BatchNorm2d-59	[-1, 64, 9, 9]	128
Dropout-60	[-1, 64, 9, 9]	0
Conv2d-61	[-1, 64, 9, 9]	36,864
BatchNorm2d-62	[-1, 64, 9, 9]	128
ReLU-63	[-1, 64, 9, 9]	0
Dropout-64	[-1, 64, 9, 9]	0
Conv2d-65	[-1, 64, 9, 9]	36,864
BatchNorm2d-66	[-1, 64, 9, 9]	128
ReLU-67	[-1, 64, 9, 9]	0
Dropout-68	[-1, 64, 9, 9]	0
AvgPool2d-69	[-1, 64, 1, 1]	0
Conv2d-70	[-1, 2, 1, 1]	128

Total params: 221,856
Trainable params: 221,856
Non-trainable params: 0

Input size (MB): 0.12
Forward/backward pass size (MB): 43.71
Params size (MB): 0.85
Estimated Total Size (MB): 44.67

Figure 32. Summary of CNN

Now, let's take a look at the activation functions in the various blocks (Figure 33):

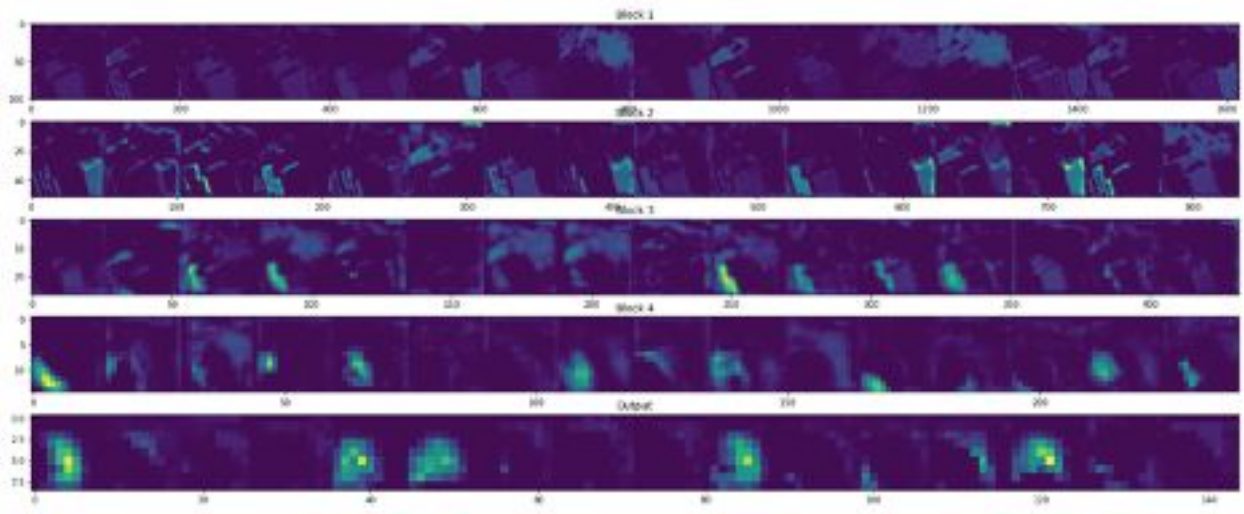

Figure 33. Activation function at various blocks

The following plots show the variation of the loss function and the accuracy as we proceeded over the 150 epochs (Figure 34).

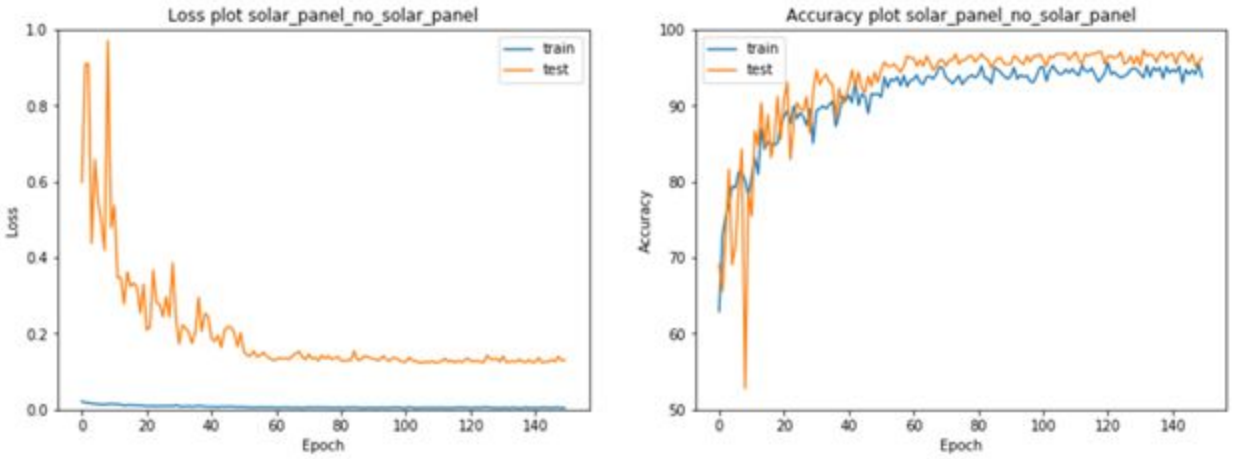

Figure 34. Loss function with increasing number of epochs.

The following are plots for the loss function on the test and train dataset individually,since the above loss function plot isn't clear.

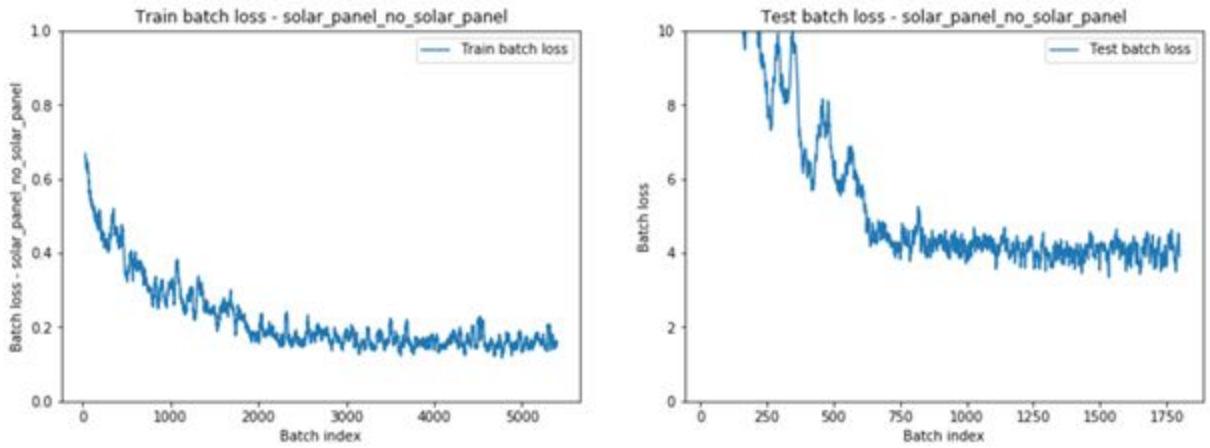

Figure 35. Loss function on training and test sets.

7. Conclusions

Having trained four different models to solve the problem of detection of solar panels on satellite images, it is evident that the convolutional neural network approach yields the best results by far. The first two models, which used simple approaches of feature extraction along KNN models, while being better than random guessing, still did not manage to produce great results. The ensemble model built on top of the first two models, using their predictions as input features along with a logistic regression model to improve the classification results did improve the AUC and AP values considerably, but it still did not manage to go above values of 0.9.

Moving on to the CNN methods, it has been quite a learning experience. We can conclusively say that we explored the Res-Net architecture. There are other state-of-the-art techniques such as Dense-Net and Inception that we could try in the future. However, we are satisfied with the results that Res-Net has given us, after being modest with the number of layers and kernels used.

8. References

[1] U.S. Solar Market Insight® (https://www.seia.org/about-solar-market-insight-report)
[2] J. Yu, Z. Wang, A. Majumdar and R. Rajagopal, “DeepSolar: A Machine Learning Framework to Efficiently Construct a Solar Deployment Database in the United States” (https://www.cell.com/joule/fulltext/S2542-4351(18)30570-1)
[3] A. J. Newell and L. D. Griffin, "Multiscale Histogram of Oriented Gradient Descriptors for Robust Character Recognition," 2011 International Conference on Document Analysis and Recognition, Beijing, 2011, pp. 1085-1089.

[4] T. I. Dhamecha, P. Sharma, R. Singh and M. Vatsa, "On Effectiveness of Histogram of Oriented Gradient Features for Visible to Near Infrared Face Matching," 2014 22nd International Conference on Pattern Recognition, Stockholm, 2014, pp. 1788-1793.
[5] K. Fukunaga and P. M. Narendra, "A Branch and Bound Algorithm for Computing k-Nearest Neighbors," in IEEE Transactions on Computers, vol. C-24, no. 7, pp. 750-753, July 1975.
[6] A. Singh, "A Detailed Guide to the Powerful SIFT Technique for Image Matching (with Python code)" (https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-python/)
[7] A. Singh, "Feature Engineering for Images: A Valuable Introduction to the HOG Feature Descriptor" (https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/?utm_source=blog&utm_medium=3-techniques-extract-features-from-image-data-machine-learning)
[8] S. Mallick, "Histogram of Oriented Gradients" (https://www.learnopencv.com/histogram-of-oriented-gradients/)
[9] J. Brownlee, "A Gentle Introduction to Batch Normalization for Deep Neural Networks" (https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/)
[10] The School of AI (https://sites.google.com/theschoolofai.in/theschoolofai/home)
[11] https://sites.google.com/theschoolofai.in/theschoolofai/home