Design and Evaluation of a Machine Vision System for Identifying Cracked PV Panels

Jerome Wynne

July 17, 2017

1 Problem Description

This work developed a system for automatically identifying cracks in PV panels from their 2D grayscale computed tomography images. 50 images were provided. 38 of these were deemed to be cracked. Several of these images are shown in Figure 1.

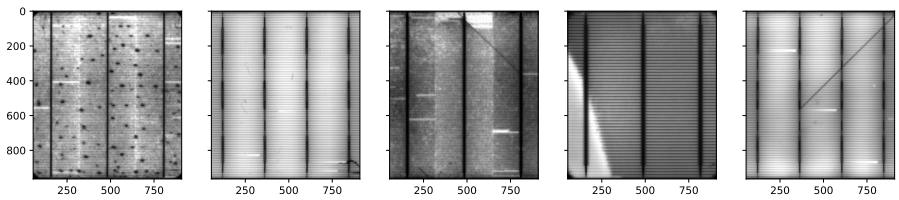


Figure 1: Scans of the PV panels that were analyzed for damage. The first, third, fourth, and fifth panels from the left contain cracks. Pock-marks similar to those seen on the first panel were not regarded as cracks.

The median frame height for these images was 965 pixels; their width was similar. Depending on the model and filters used, the images were shrunk for processing.

As can be deduced from Figure 2, it was sometimes difficult to ascertain whether a given mark was a crack or a scratch. For the purposes of this work, the latter were considered to be fainter and wigglier - heuristic measures in the absence of any conveniently quantifiable differences. The labels themselves consisted of manually drawn binary masks. Each pixel in a mask was in effect an indicator representing whether the associated image pixel was part of a crack. The masks created were validated by a panel inspector. The centers of thick cracks were not labelled, on the basis that they would make learning difficult if small kernel sizes were to be used. For example, a 20×20 window centered on a thick crack would be a dark array almost indistinguishable from other dark non-crack regions in the images (in the absence of positional information, at least).

Observed sources of noise in the images were as follows:

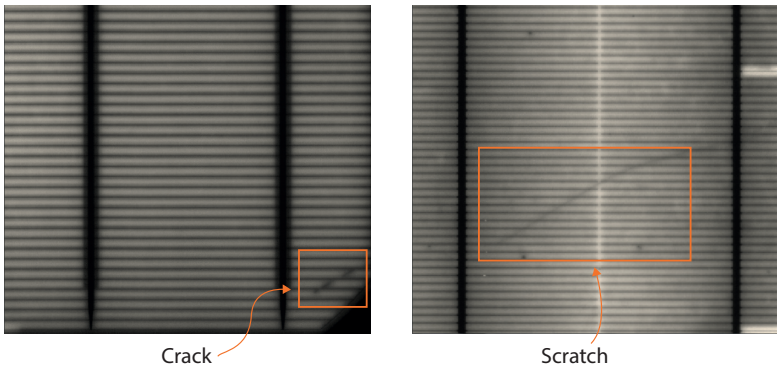


Figure 2: Distinguishing between cracks and scratches was difficult in certain cases, making ground truth labels somewhat subjective.

- Dead cells (cells that were unusually bright relative to their neighbors)
- Scratches and marks (dark lines)
- Debris in the scanner (dark patches)
- Hail damage (dark pock-marks)
- Variation in the panel model (variation in the panel's layout)
- Delamination (low-frequency intensity variation)

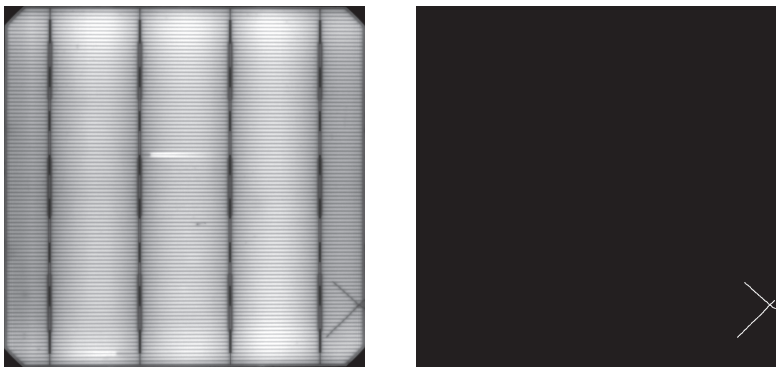


Figure 3: An image of a panel and its associated set of labels.

The ideal panel classifier was defined as being one that:

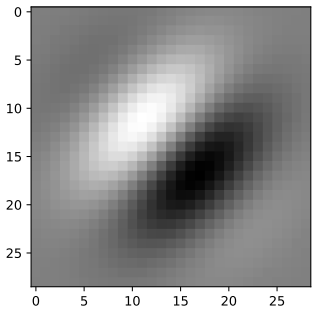


Figure 4: The real component of the two Gabor filters applied to the images. The other filter was equivalent but flipped across the vertical axis.

- Provided accurate classification of panels according to the classes cracked/not cracked.
- Provided estimates of class membership probabilities to allow operators to tune the model's sensitivity.
- Provided per-pixel classification to enable an operator to quickly the area flagged as being cracked.
- Operated within reasonable time and compute constraints.

Interpretability was not considered a priority.

1.1 Pre-Processing

Various filters were applied to the images in an attempt to emphasize cracks and suppress other features. Horizontal differencing of pixel intensities proved to be an effective means of filtering the horizontal cells. A 3×3 Laplace filter was used to achieve this effect - a Prewitt filter was also tested, but yielded basically identical results. A large proportion of the cracks were found to propagate diagonally, an attribute that was exploited by applying a pair of Gabor filters to the image then taking the magnitude of the difference between their real components. The Gabor kernel used is shown in Figure 4. By combining horizontal differencing and the described Gabor filtering, the diagonal cracks were substantially enhanced relative to the rest of the image, with the exception of the panel's corners. This process and its results are shown in **??**.

As an alternative method for extracting information pertaining to edge orientation, histograms of oriented gradients were used. An image's gradients were estimated using a combination of Laplace filters, then their magnitudes were accumulated in bins corresponding to distinct orientations. The bin counts in each cell were normalized relative to neighboring cells, then the cell histograms were flattened and concatenated to be fed into classifiers that accepted vector inputs. So far, HOGs have only been tested against

classifiers accepting vectors: one possible line of enquiry is to pass them to classifiers accepting tensors of higher orders. Aside from HOGs, the Hough transform may also be a promising candidate for identifying diagonal lines that do not correspond to the panel's corners. It has yet to be trialled.

1.2 Modeling

The models trialled so far include:

- Generative models against pixel intensity (i.e. modeling local pixel intensities using a multivariate normal distribution)
  - These were also tested as filters, the output of which was fed to a nonlinear classifier. An example of this sequence is shown in Figure 5.
- Canned classifiers from `sklearn` - random forest, radial-basis function support vector machines, logistic regression.
- Shallow convolutional neural networks against pixel intensity.

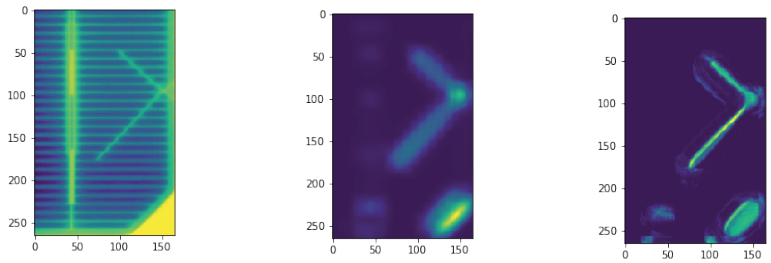
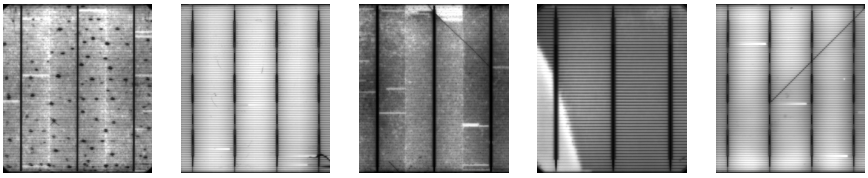


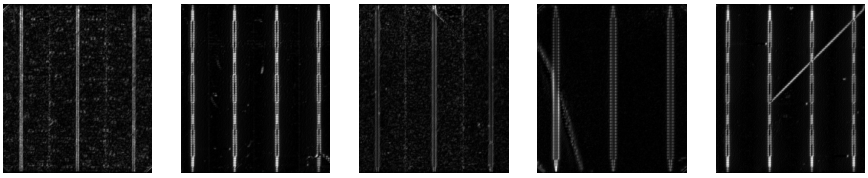
Figure 5: Left: raw image. Center: Log-odds returned by a normal model fit to 20×20 patches of cracks. Right: Crack probabilities returned by a random forest fit to the log-odds returned by the normal model. (This needs a colorbar!)

TBC

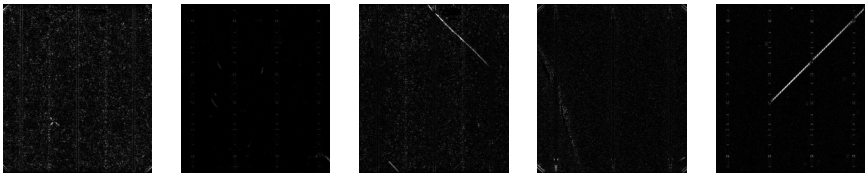


(a) Raw images.

(b) Vertical Sobel filter.



(c) Vertical Sobel filtering followed by a Gabor filter.



(d) Vertical Sobel filtering followed by a difference of two Gabor-filtered images.

(e) Original image masks (i.e. ground truth).

Figure 6

Placement Update

Last time we met

- 3rd July (2 weeks ago)
- I presented:
  - The labelling strategy I have been using (i.e. per-pixel binary masks).
  - An image subsetting and augmentation procedure.
  - Canned classifiers against raw pixel intensities.
  - Canned classifiers against vectorized data of the prominence of intensity gradients in certain directions (i.e. Histograms of oriented gradients).
  - A proposed scheme for making adjacent pixel labels more consistent with one another (more on this later).
  - Problems w.r.t. splitting the data and obtaining robust classification accuracy estimates (i.e. the dataset is small).
- We agreed I should expand the feature space, *by any means necessary.*

What I've been up to since then

- Spent a fantastic couple of weeks in Barbados.

Figure 7: My co-working space.

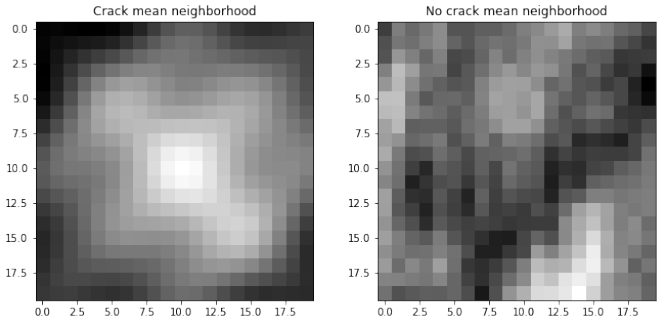
- Time split: 25% creating new image representations, 30% reading about modelling strategies, 10% building models, 35% building testing infrastructure.
- Other image representations:
  - Gabor filters and other edge/ridge-based filters.
  - Global fourier transform.
  - Local fourier transform.
  - Hough transform.

  - Local binary patterns (more on these later).
  - Hierarchical filtering (*a la* convnets).

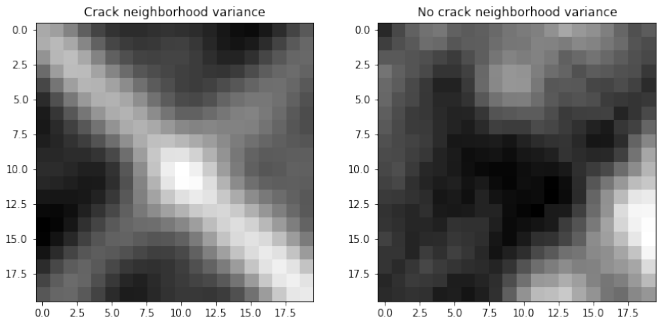
- Modelling strategies: relaxation labelling and Markov random fields.
  - Description image.
  - Performance depends on having an initial classifier that is - at minimum - passable (obviously).
  - Yields an accuracy increase of several per cent.
  - Disagreement regarding choice of compatibility function and update equations (former: transition probabilities, correlations, mutual information).
  - Markov random field model: might be more appropriate/effective?

- Model implementation: simple convolutional neural network.
  - Initial performance suggests an accuracy of c. 85% against a balanced set of 20x20 image patches.
  - Against genuine images, where approx. ¡ 0.1% of patches correspond to cracks, this translates into poorer performance.
  - Plan to experiment with:
    * Other patch sizes.
    * Increasing the number of input channels (e.g. filtered version of the patch, rescaling the entire image down to the patch's size, providing a map marking the patch's position within the image and so on).
    * Using spatial transformer networks (learning a transformation mapping the input to a ''canonical" (i.e. standardized) representation of a crack) to improve classification accuracy and act as an attention mechanism (to reduce the fraction of an input image that needs to be processed).
    * Making the network deeper (it's only two layers at the moment).
    * Implementing filters mid-network (e.g. applying a Hough transform to the network's output, then feeding that into a multilayer perceptron).

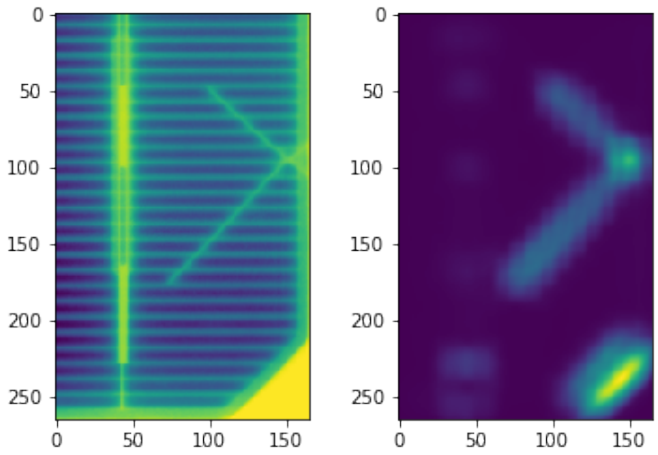
- Model implementation: Gaussian discriminant analyzer.
  - Need to make it compatible with the testing API.
  - It seemed to do okay - needs to be plugged into the new testing framework.
  - Testing infrastructure:
    * All in TensorFlow.
    * Separates preprocessing and modelling.
    * Provides a consistent modelling environment, making it easier to form ensembles or chain models later.



(a) Group means (20×20 patch).



(b) Diagonal entries of group covariance matrices.



(a) Query image. (b) Log-odds of query image.

* Standardizes performance reporting and allows models to be visualized (court oisie de TensorBoard).
* Easily extensible to larger datasets.
* Ensures the data splitting procedure is transparent and consistent.
* Demo.

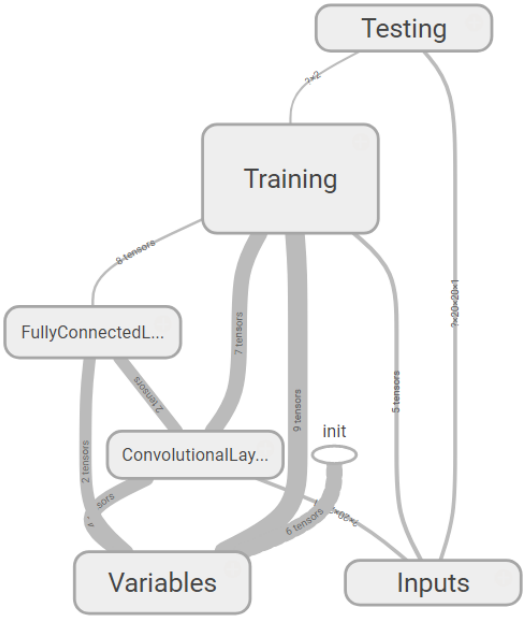


Figure 10: Graph of the convolutional network that has been implemented, courtesy of TensorBoard.

Figure 11: Using TensorBoard we can create standardized reports, which should make evaluating models more straightforward.

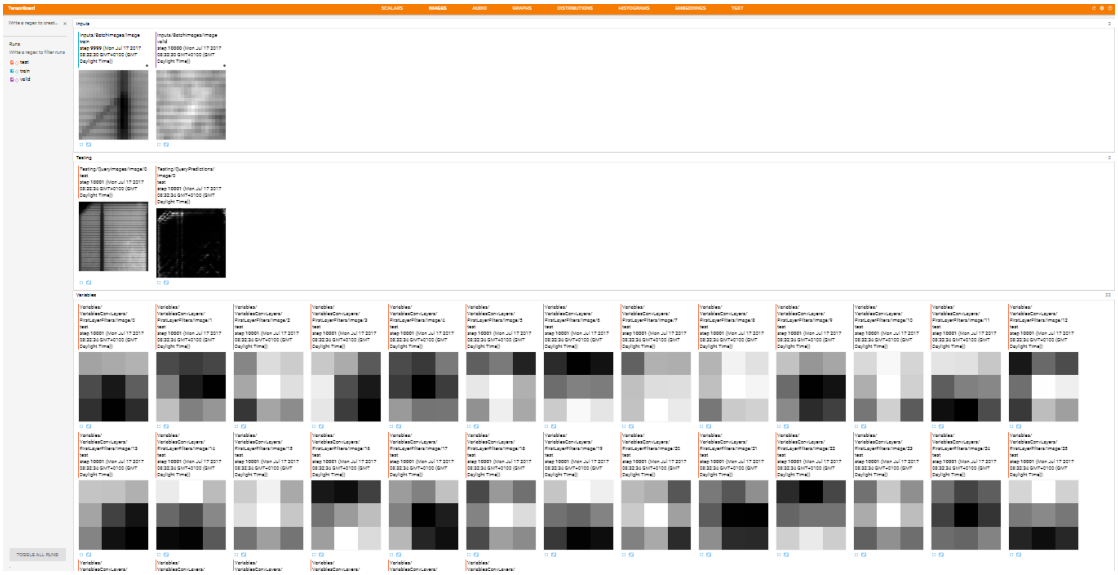


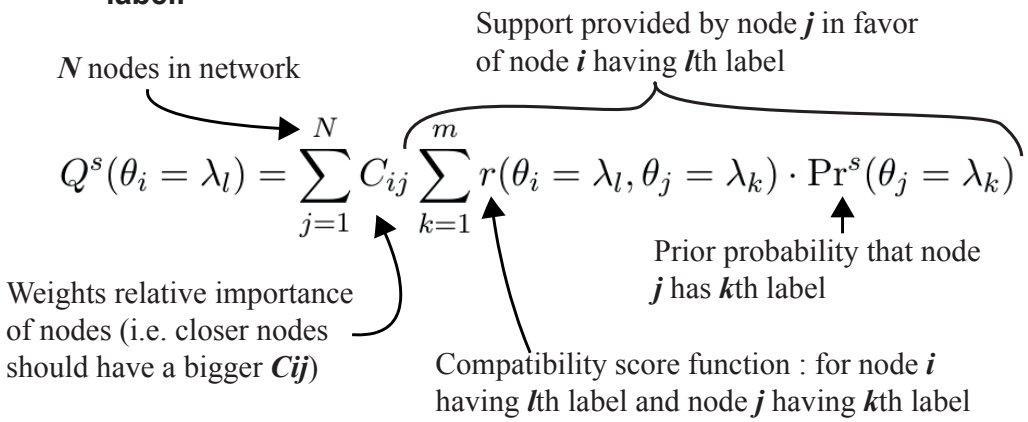
Figure 12: A sample of the filters learnt by the first layer. These may be useful for preprocessing images for other classifiers.

Problems I have and what I plan to do about them

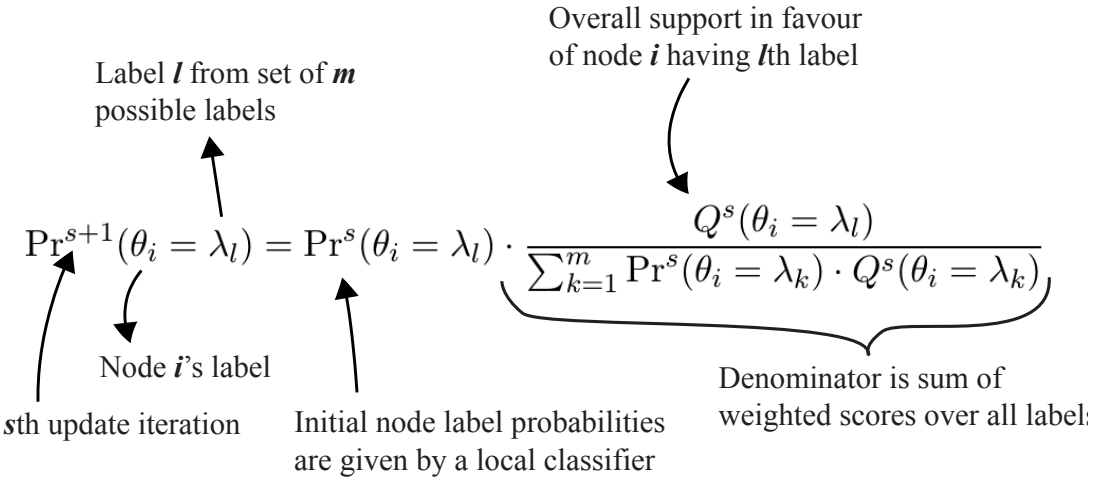
- Data:
  - Splitting the dataset after sampling patches from the images results in isomorphic training and validation datasets: The dataset must be split at the full-image level for our performance measures to be valid.
  - Because the dataset is so small, the splitting is very granular. Limited representation of rare crack types in the data available means that the validation and training datasets contain different distributions of crack types.
  - Consequently, model fit and accuracy metrics are both very sensitive to the split chosen. If the training data contains all rare cracks, the accuracy will appear high. If it contains unusual cracks, a model will appear to perform badly.
  - If this problem afflicts all models equally, then comparisons of performance between models are valid. If not, then we cross-validation is necessary. This is time-consuming; additionally, comparing models becomes more subjective.
  - If you have Azure access, please could I have a copy of the additional data?
- Middle of placement:
  - Wary of reaching week 8 with lots of half-finished, half-evaluated models.
  - Intend to focus on:
    - Convolutional neural networks

Relaxation Labelling: An Overview

Idea 1: Quantify the contextual support in favor of a node having a given label.



Idea 2: Iteratively reweight a node's label probabilities in favor of labels that have lots of contextual support.



Key idea: Large support Q for a node having a particular class => Higher probability of that class for the node => Improved labelling consistency.

* The compatibility and update equations are heuristic (i.e. they work). Efforts towards a Bayesian method are ongoing (in 1997).

    - Markov random fields.
  - Current objective: solve the PV panel crack identification problem, write a report summarizing the methods applied and their efficacy..

- Feedback:
  - Pointers/warnings?
  - Suggestions for making sure I produce something fruitful? (What would *you* find useful for me to deliver at the end of this placement? Report, model, code?)
  - Witticisms?
  - Presenting style / communication skills?