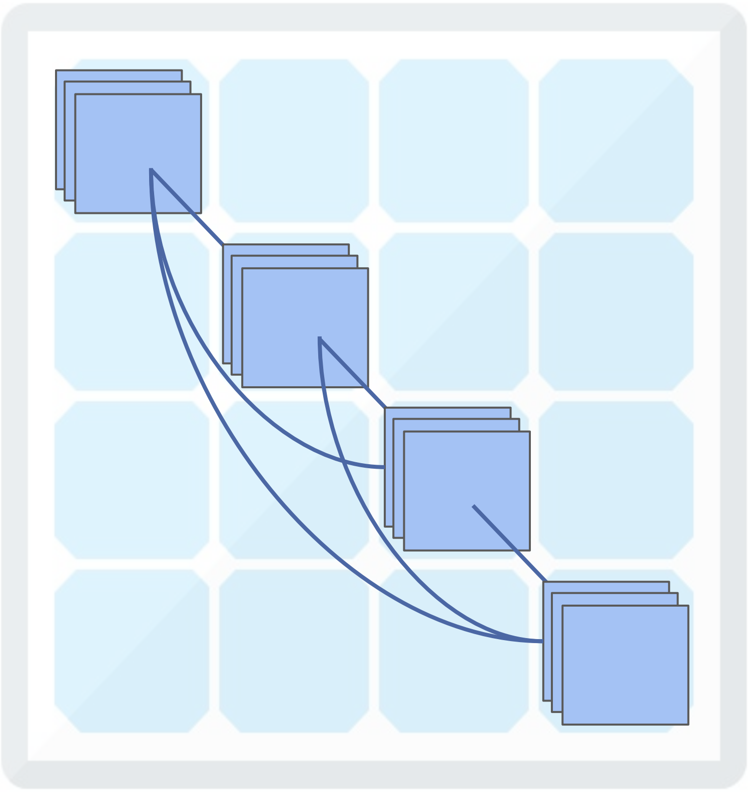
DenseNet-based model for Defect Classification on Solar Panels



Exam project for

Computer Vision

Tuesday, 28th February 2023

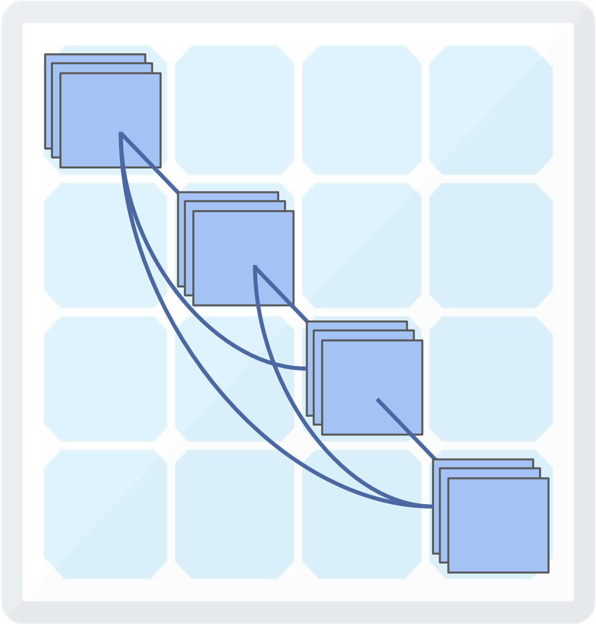
powered by

Simone Pio Caronia

Paolo Ruggirello

Agenda

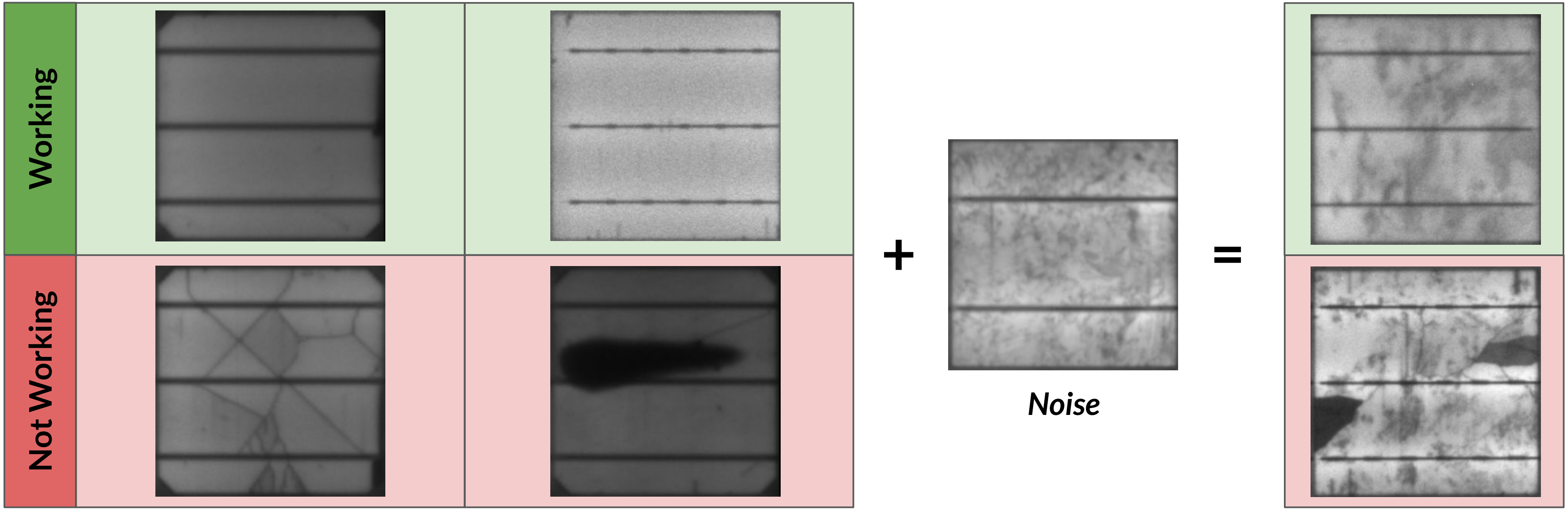
1. Look at images
2. Preprocessing
3. Model experiments
4. Transfer learning - **DenseNet**
5. Final complete model
6. How we improved performances
7. Results



1. Look at images

In the full dataset of 2624 images, there are:

- 1803 **WORKING** cells
- 821 **NOT WORKING** cells

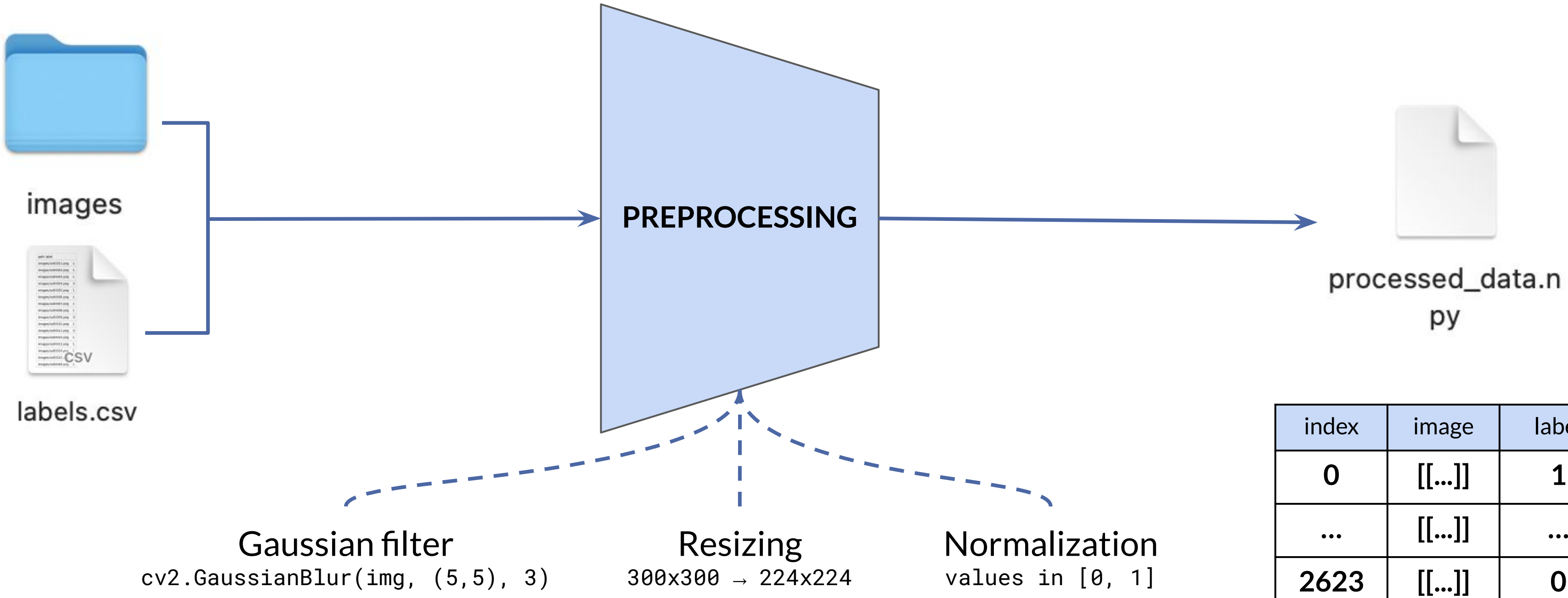


Cracks

No reflection

2. Preprocessing

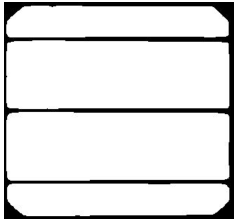
The preprocessing pipeline implemented gives in output a unique file with both transformed images and related labels.



index	image	label
0	[[...]]	1
...	[[...]]	...
2623	[[...]]	0

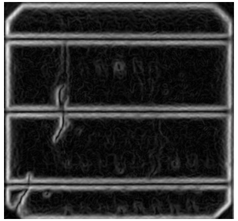
2.1 Failed preprocessing

Different preprocessing pipelines have been tried, but the metrics were lower than the ones got with the final model and preprocessing chosen.



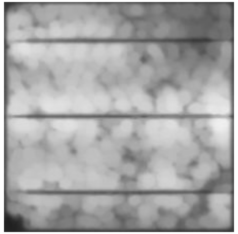
1. *Mask*

Applied the mask, calculated as the intersection of link lines of solar cells.



2. *Gradient*

Applied Sobel/Laplacian filters to try working on it.



3. *Morphological operations*

Applied opening/closing operations to remove small objects.

And much more:

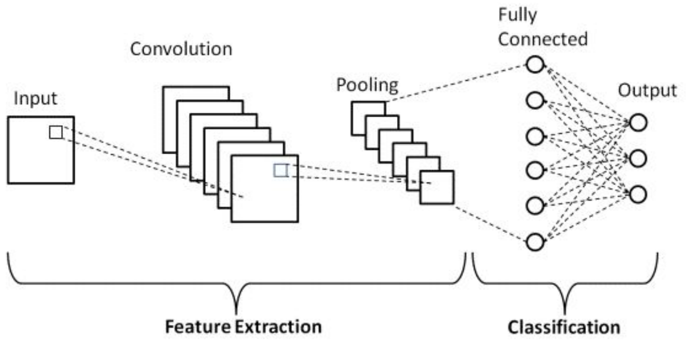
- LBP;
- Reading images in RGB;
- Seam carving;
- Rescale intensity;
- Thresholding: Global, Otsu method, Variable local;
- Custom 3-channels image;
- Sharpening filters;

and combinations of above.

3. Model experiments

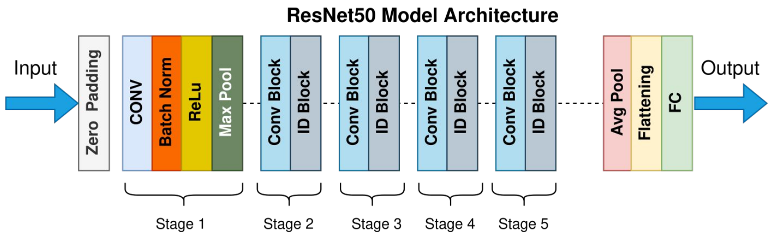
Before getting to the final model, different approaches were tried. The most relevant ones are:

Homemade CNN



DISCARDED
Too shallow to extract good features, low metrics, time consuming

ResNet-50



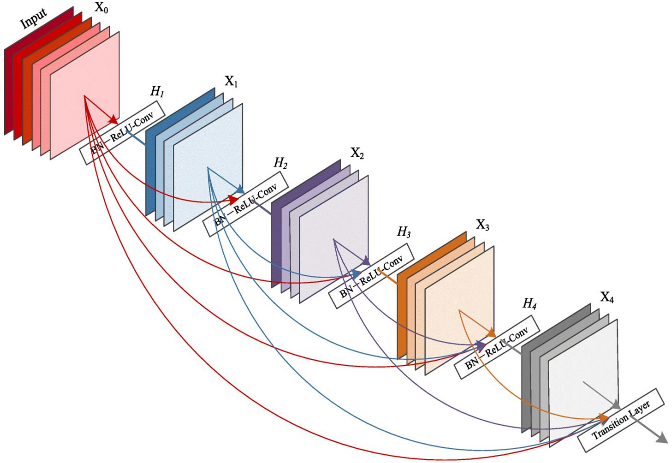
DISCARDED
Better features, short training time, but low metrics

4. Transfer learning - DenseNet

The approach chosen is based on transfer learning.

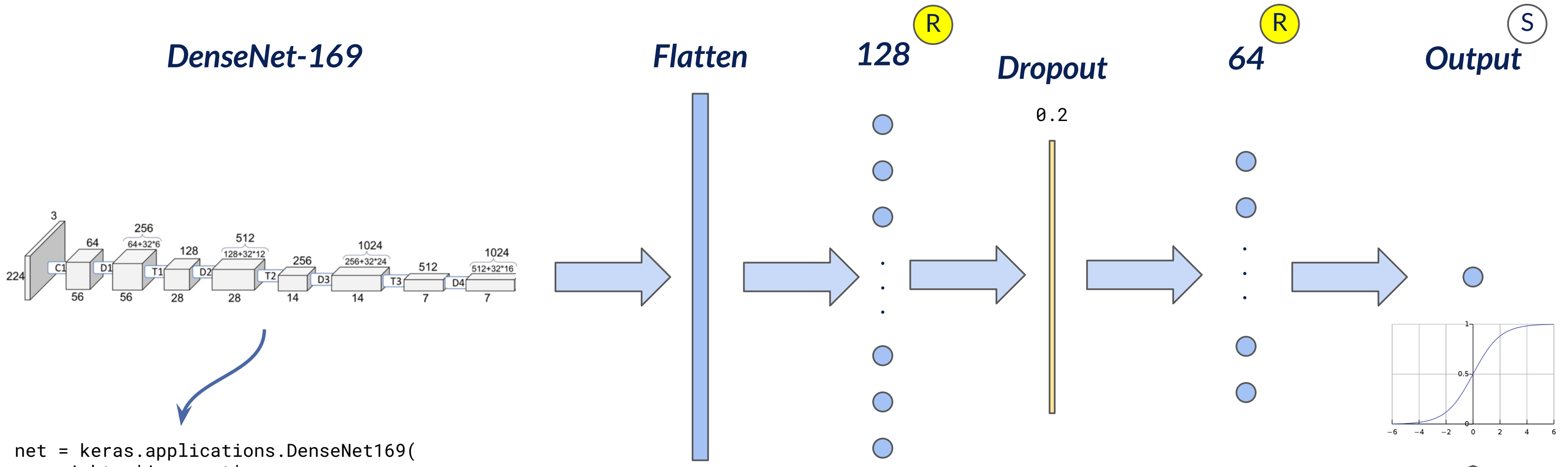
The base model selected is **DenseNet**:

- Obtained the best results related to handcrafted version;
- Transfer learning using pretrained model on ImageNet;
- Exploited its depth to extract better features;
- Skip connection as FEATURE MAPS CONCATENATION.



Thanks to that, feature maps extracted in first layers are propagated to all the next ones. Those FM contain information about local characteristics.

5. Final complete model



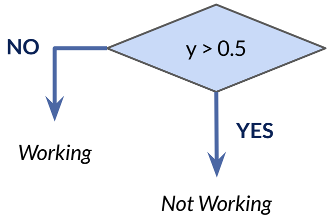

```
net = keras.applications.DenseNet169(
   weights='imagenet',
   include_top=False,
   input_shape=(224, 224, 3)
)

net.trainable = False
```

Legend:

R ReLu

S Sigmoid



6. How we improved performances

In order to improve performances, the following elements were tuned for the training phase:

- **Early Stopping**: regularization method to avoid overfitting;

```
keras.callbacks.EarlyStopping(
    monitor='val_loss',
    mode='min',
    patience=5,
    restore_best_weights=True)
```

- **Optimizer**: Adam optimizer, evolution of stochastic gradient descent which perform well in computer vision tasks;

```
keras.optimizers.Adam()
```

- **Class_weight**: used to give more weight to class 1 (Not Working) to improve F1 score.

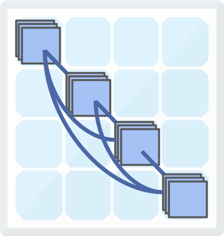
- **Data augmentation**: `np.flipud() | np.fliplr()`

Other hyperparameters:

- **Epochs**: 20;
- **Batch size**: 32;
- **Loss function**: `'binary_cross_entropy'`.

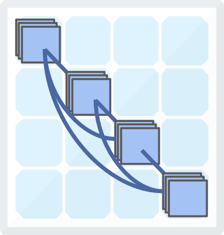
7 Results per fold

Fold	Accuracy	F1 score
1	83%	69%
2	84%	73%
3	84%	70%
4	85%	72%
5	85%	71%
6	85%	74%
7	82%	70%
8	84%	74%
9	85%	73%
10	83%	72%



7.1 Results

Accuracy	F1	St. Dev. Accuracy
84%	**72%**	**0,01**



Thanks for your attention!

Now, Q & A

Simone Pio & Paolo