Data set

In this task we worked with a flowers dataset that was collected in Volkani institute.
The data includes 472 cropped images of flowers and non-flowers, with corresponding labels.
We loaded the images and resized them according to 224*224*3 scale.

We split the Data accordingly:

- The first 300 cropped images were used for training set.
    - We split the training set to 20% validation set and 80% training set.
- The remaining 172 images were used for test set on which the results were obtained.

Basic pipe

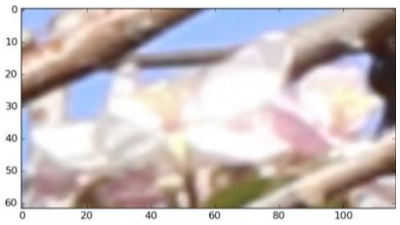
We Built a flower classifier with task transfer from a pre-trained network (ResNet50V2).

The classification is flower or not flower. Thus, the output layer had a single neuron estimating p^(y=flower|x). We used the pre-trained weights of network but we didn't use the fully-connected 1000-neurons layer at the top of the ResNet50V2 network. Instead, the layer before last H^{L-1}, (7×7×2048) was globally pooled (average pooling) over the 7×7 spatial dimensions to get a vector V of 2048 elements. Then, the output neuron p for our net was obtained as $p = \sigma(w_L \cdot V - b_L))$ with $\sigma(x) = \frac{1}{1+\exp(-x)}$.

In addition, we fine-tuned the weights of the final layers we added.

The error rate obtained over the test set was **31.4%**.

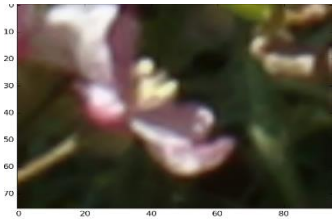
<u>5 worst errors of type 1 on the test set:</u>



Error type: 1

CNN score: 0.11111599

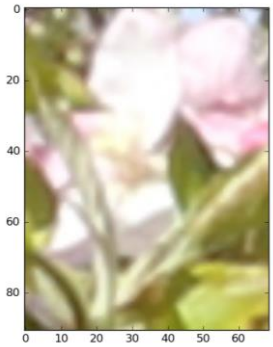
Error index: 1



Error type: 1

CNN score: 0.20069975

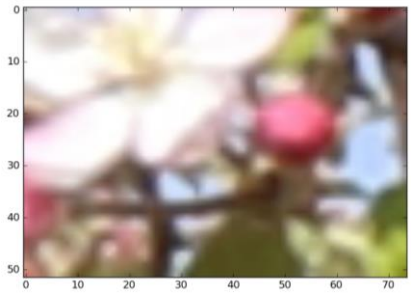
Error index: 2



Error type: 1

CNN score: 0.24023113

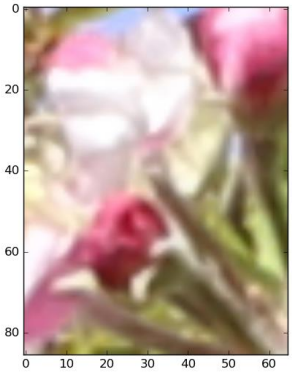
Error index: 3



Error type: 1

CNN score: 0.29645333

Error index: 4

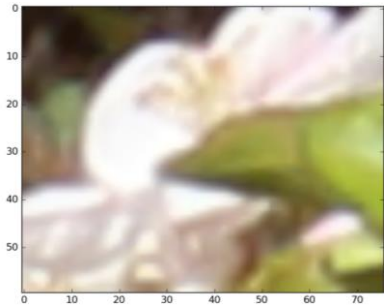


Error type: 1

CNN score: 0.40467727

Error index: 5

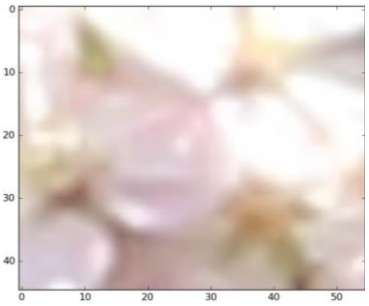
5 worst errors of type 2 on the test set:



Error type: 2

CNN score: 0.99750316

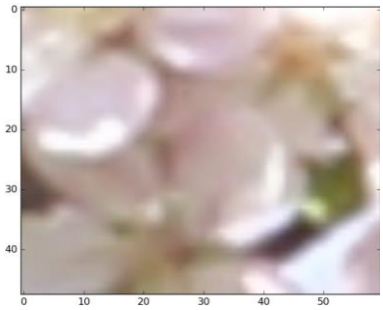
Error index: 1



Error type: 2

CNN score: 0.9969544

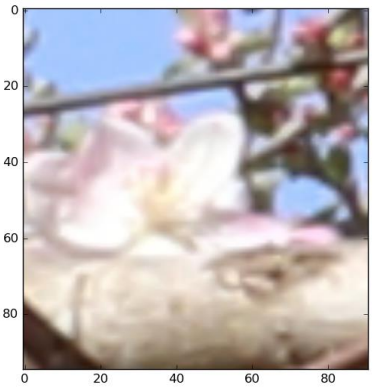
Error index: 2



Error type: 2

CNN score: 0.99674726

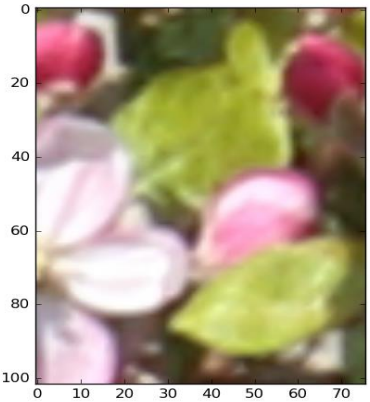
Error index: 3



Error type: 2

CNN score: 0.99644095

Error index: 4

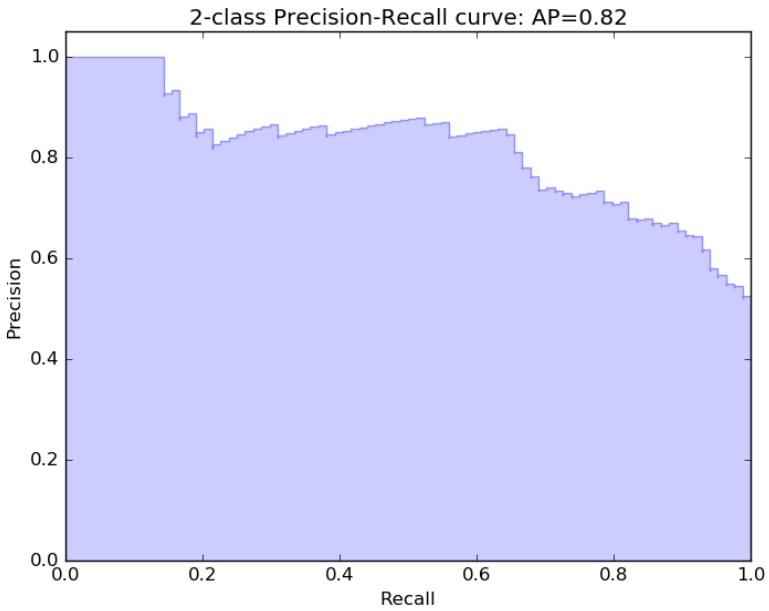


Error type: 2

CNN score: 0.99131244

Error index: 5

Precision-Recall graph:



Improvement

We tried to improve the network using:

- **Data augmentation:**
  We applied these augmentation parameters on the training set:
  • Rotation: 30 degrees
  • Width shift: 0.2
  • Height shift: 0.2
  • Shear: 30 degrees
  • Zoom: 0.2
  • Horizontal flip: True
  • Vertical flip: True
  We chose these specific parameters in order to maximize the diversity in the data but with consideration of the real data set.
  The base model with augmentation didn't had better score from the base model itself.
- **Weight decay**: We tried several options. (explanation in the hyper tuning section)
- **Dropout:** We tried several options. (explanation in the hyper tuning section)
- **Training the N-last layers.**
  We freeze all the layers except the last N layers that we trained.
  We tried different values of N, we started with N=2, and we found out that N=10 had the best results.
- **We tried different optimizers (**Adam, RMSprop, SGD) and found out that Adam had the best results.
- **Combinations of the above.**

Best Improvement

We have got the best score by changing the architecture and mixing some of the options mentioned above:

We chose to train the 10 last layers by freezing all the layers except the last 10 layers, we applied data augmentation, in addition we add dropout to the last dense layer. We add global average pooling layer and another fully connected layer of N neurons with weight decay before the last layer (the prediction layer).

The Dropout's percentage, number of neurons in the fully connected layer we added, and weight decay value were tuned separately.

Hyper-Parameters Tuning

Due to the large number of options, we ran the algorithm on a number of selected combinations. Each time, we searched for the optimal value on different variable until convergence.

The hyper-parameters options:

- Dropout: [0.5, 0.2, 0]
- Number of neurons: [500, 1000, 2000]
- Regularizer: [0,0.01, 0.05]

We divided the training set (300 images) into a training set and a validation set at a ratio of 0.8 and 0.2, respectively, of the total training set (240 images for training and 60 validation images).

For each network, we ran 20 epochs and summarize (for the five highest-accuracy scores, out of the 20 epochs) the median accuracy and the weighted average of the number of their epochs. We did this to avoid over-fitting.

Below is an output of some of the hyper-parameter tuning process:

	dropout	regularizer	FC_neurons	accuracy	best amount of epochs
0	0.5	0.01	500	0.833333	9.730159
1	0.2	0.01	500	0.833333	8.984496
2	0.0	0.01	500	0.833333	8.928854
3	0.5	0.05	500	0.833333	8.774704
4	0.5	0.01	500	0.816667	13.980000
5	0.5	0.00	500	0.816667	8.647773
6	0.5	0.01	500	0.866667	13.329502
7	0.5	0.01	1000	0.866667	8.216730

The parameters which had the best results were:

Regularizer: 0.05

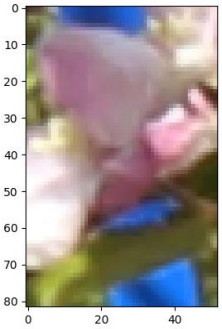
Dropout: 0.5

Number of neuron: 1000

Best Improvement- Results

The error rate obtained over the test set was 19.7674% .

5 worst errors of type 1 on the test set:



Error type: 1

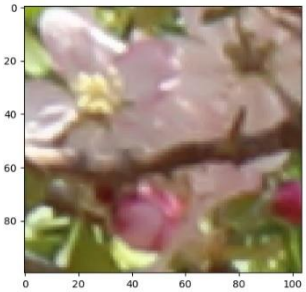
CNN score: 0.0

Error index: 1

Error type: 1

CNN score: 0.0

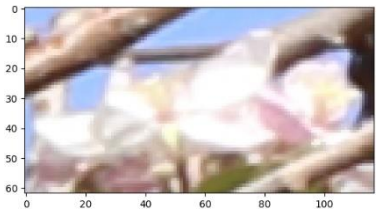
Error index: 2



Error type: 1

CNN score: 0.0

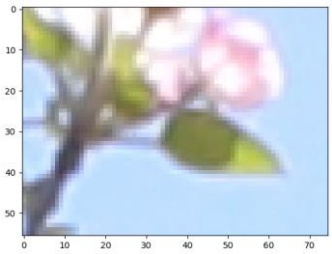
Error index: 3



Error type: 1

CNN score: 0.0

Error index: 4

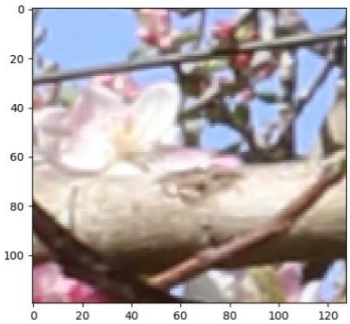


Error type: 1

CNN score: 0.0

Error index: 5

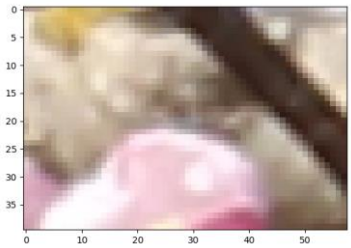
5 worst errors of type 2 on the test set:



Error type: 2

CNN score: 1.0

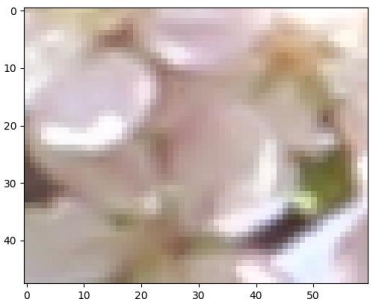
Error index: 1



Error type: 2

CNN score: 1.0

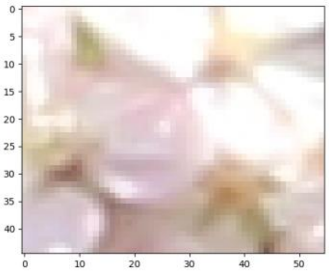
Error index: 2



Error type: 2

CNN score: 1.0

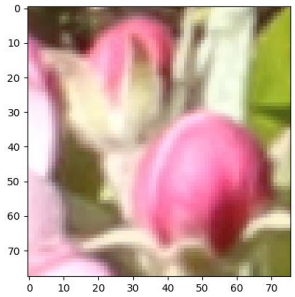
Error index: 3



Error type: 2

CNN score: 1.0

Error index: 4

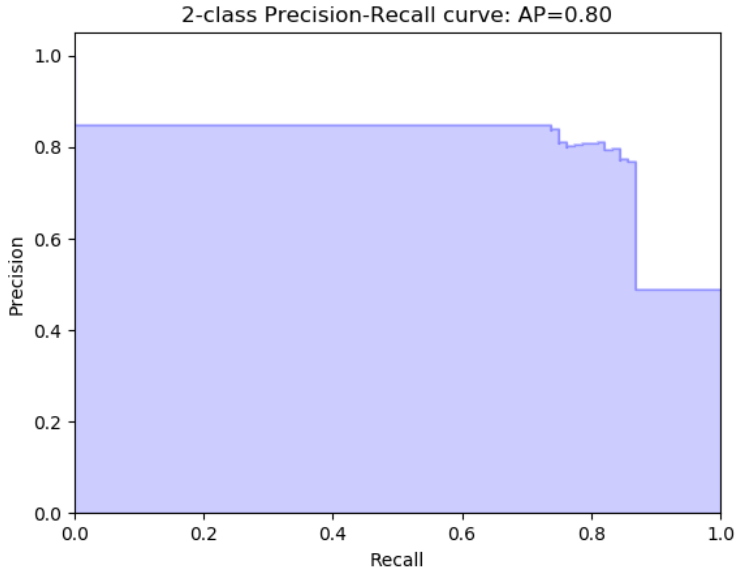


Error type: 2

CNN score: 1.0

Error index: 5

Precision-Recall graph:



We improved the accuracy of the base model by 11.67 percentage.