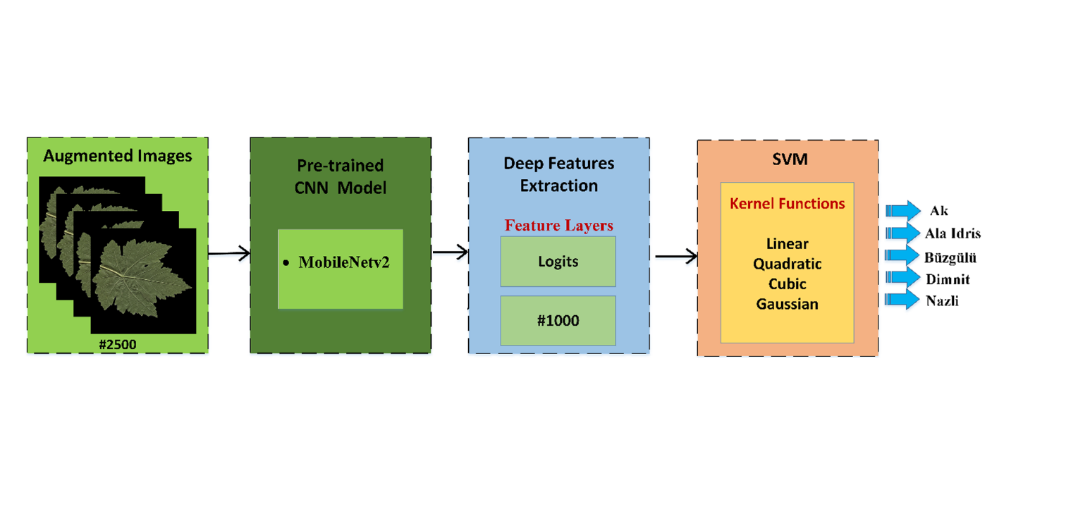
FirstName LastName

Data Mining

Final Project

Description

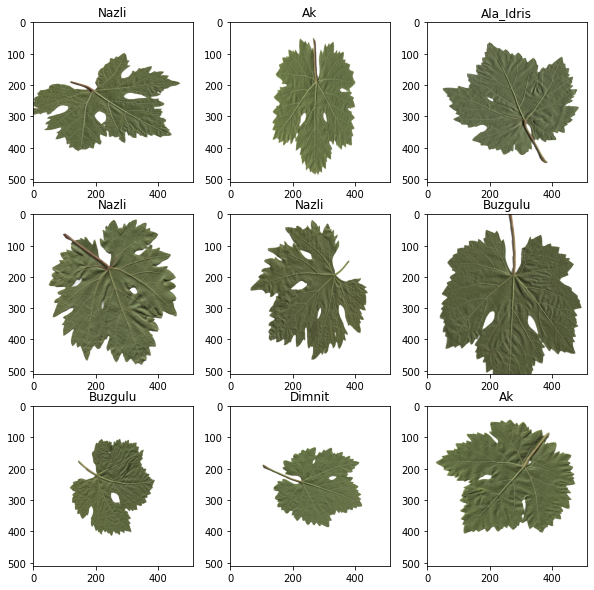
According to the article mentioned in the project description the model is consisted of a CNN part used for feature extraction and a SVM classifier which uses the CNN(MobileNetv2) extracted features, the figure below shows the idea here.

Process

In this project, we’ll augment the size of our training set (after splitting the data into test and train) by 4 times, at the end we’ll have 1500 training samples.

The augmenting section uses 3 different augmentation methods:

* Horizontal flipping
* Scaling with range of (0.75 to 1.25)
* Random rotations

Here’s a peak of our augmented training set:

We will use the pre-trained CNN (MobileNetV2) and we extract the features of each sample using n features, further a SVM Classifier will be trained using these features to be able to detect the class of each leaf, we’ll then use the same process of feature extraction and classifying to validate our test dataset.

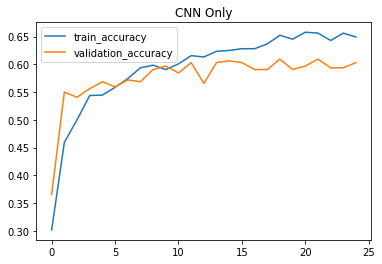
To load the model *tensorflow\_hub* library is required.

The SVM-Classifier is build using *sklearn* library.

**CNN-Only**

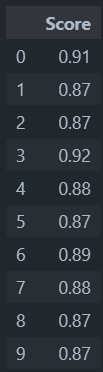
Before we delve into the main approach we’ll try vanilla-CNN without the help of any classifier such as SVMs, there network architecture is just the mobileNetV2 plus a flatten layer and 2 dense layers which leads to a final output layer of size 5 and softmax activation function.

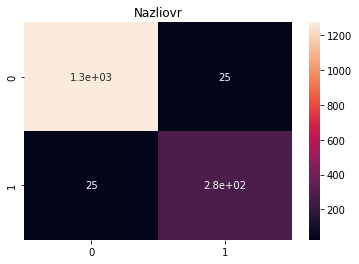
Here's the mean results of running this model for seed 1 to 10:

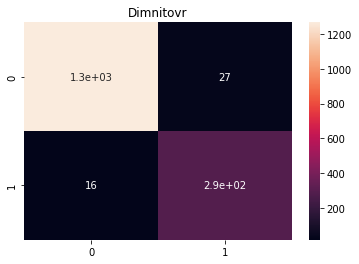
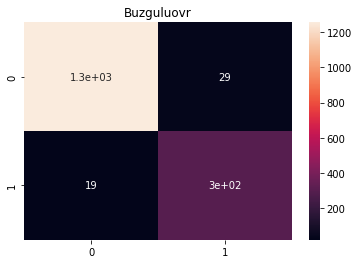
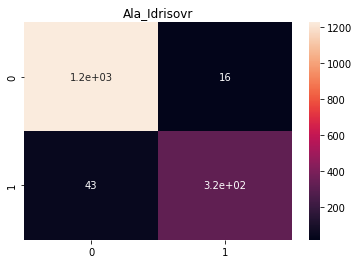
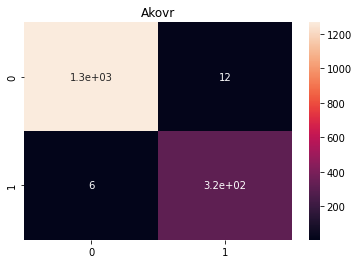


The 10fold-crossValidation methods will be covered in the next model which combines the power of CNN’s feature extraction and SVM’s classifying advantages.

Also notable to say that MobileNetV2 uses an input shape of 224 \* 224 which is not the same as out inputs (511 \*511) so we’ll need to simply resize our picture to match the desired dimensions (various methods are available by TensorFlow for that cause, the default is bi-linear)

Using this technique we’ll be having 88.28571 percent accuracy which is the mean of the 10 CV runs, here is the detailed report of the CV scores:

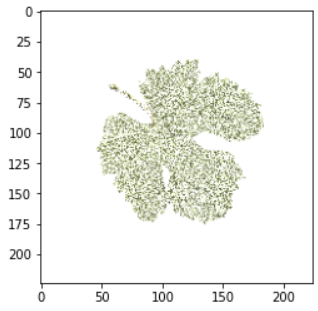
Also as we used the One vs Rest method in our training, this the heat map of every class in the SVM training process; the ovr simply means that we’ll run the SVM model one time for each class, every time the confusion matrix would show us how many of the e.g. Nazliovr Class was predicted correctly and how many time it was predicted as the “rest” of the classes, you can see the heatmap of each class below:



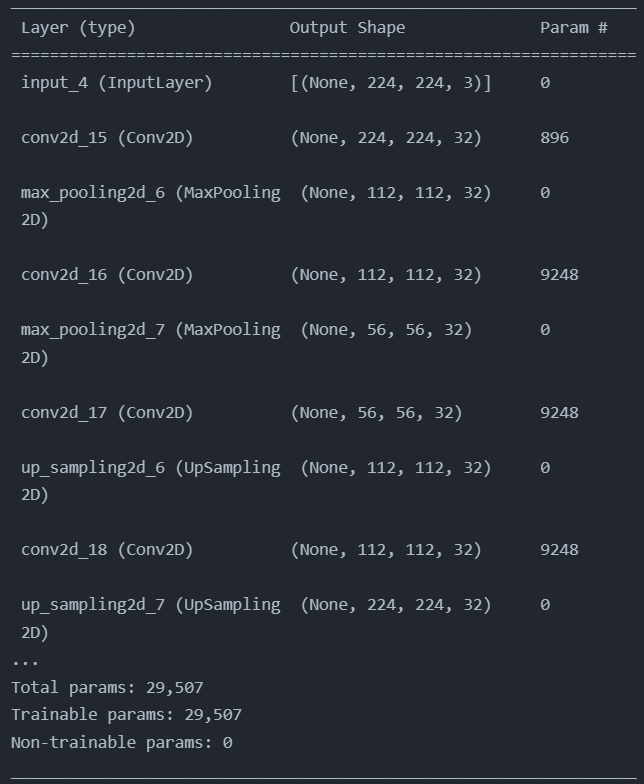
**Auto-Encoder**

We’ll add some noise in every picture of our dataset using a gaussian distribution and then feed the noisy pictures to an auto encoder which we’ll expect to yields us the clear ones.

The Noisy pictures look something like this:



the summary of the model is shown below:



**Conclusion**

Having a CNN could greatly benefits us as it could extract complex 2D features that other methods cannot, though classifying using a VannilaCNN is not suggested as it can only simulate something like a non-linear regression model, though other Classifiers such as SVMs could help the process to get more precise.

p.s. didn’t have time to fine-tune the auto encoder. Though the architecture should work just fine.