50.039 Theory and Practice of Deep Learning HW1
Loh De Rong (1003557)

Pre-processing

There are 1145 images in total. They are split into 65% for training, 15% for validation, 20% for test, stratified by their class labels to ensure proportionate class representation for every split.

Data split shape	**Sample size per class (real index → new index)**		
	Spring (9 → 0)	**Summer (10 → 1)**	**Autumn (11 → 2)**
X_train: (744, 1024)	68	577	99
y_train: (744,)			
X_val: (172, 1024)	16	133	23
y_val: (172,)			
X_test: (229, 1024)	21	177	31
y_test: (229,)			

Table 1. Data splitting into train-val-test sets, stratified by class.

Hyperparameter Tuning

For this report, we are only required to finetune the regularization parameter c for the linear kernel. In the table below, I have also included other kernels in the experiment for comparison.

Each model is trained on the training set, and its performance is measured on validation set using the class-wise average accuracy formula.

Kernel	**Regularization parameter c**					
	0.01	**0.1**	**$0.1^{0.5}$**	**1**	**$10^{0.5}$**	**10**
Linear	0.50293	0.47011	0.48460	0.48460	0.48460	0.47011
Poly	0.53074	0.53074	0.53074	0.50175	0.50426	0.52760
RBF	0.55408	0.55408	0.55408	0.55909	0.54460	0.54209

Table 2. Hyperparameter Tuning for kernel type and regularization parameter.

For the linear kernel, the best regularization parameter constant is **0.01**.

Analysis of the performance between kernels

Comparing between kernels, as we hold the value of c constant, it can be observed from Table 2 that their performance is always in the following decreasing order: RBF < Poly < Linear. This is likely because the dataset is not linearly separable, so a non-linear kernel would perform better in transforming the data to a higher dimension that is more linearly separable. In this view, the RBF kernel is therefore the best out of all 3 kernels as it is able to project the data into an infinite higher dimensional space where the data becomes linearly separable.

Performance measures on validation and test sets

Using the best regularization constant = 0.01 found earlier for linear kernel, the performance of each classifier is reported for both validation and test sets, when trained on train set only or on train + validation set.

Both measures of accuracy provide us different insights into the performance of the classifier.

- The vanilla accuracy simply measures the percentage of predicted labels that match exactly the corresponding true labels. It is a naive way of measuring the performance of a model.
- The class-wise average accuracy measures the average percentage of the number of correctly predicted positive samples over the total number of true positives for all 3 classes. The sample size of each class is different, so using this accuracy ensures that each class is equally represented, hence the result is not biased toward the performance of any class.

Vanilla Accuracy

	Trained on train set only	**Trained on train + val set**
Validation score	0.83140	0.90698
Test score	0.82096	0.83406

Table 3. Vanilla accuracy for linear SVC (c = 0.01)

Class-wise average accuracy

	Trained on train set only	**Trained on train + val set**
Validation score	0.50293	0.71739
Test score	0.50620	0.52959

Table 4. Class-wise average accuracy for linear SVC (c = 0.01)

The results make sense because when the model is trained on train + val set, as opposed to being trained on train set only, the validation and test scores increase. The model has more labelled data points to learn from, and thus performs slightly better.

The reason why the validation score increases significantly more is because the model is trained on validation set as well. The data from the validation set is not unseen in this case, and hence, it is not surprising that model is able to predict the labels well for the images it has been trained on.

Finally, when trained on train set only, both the validation and test scores are very close, indicating that the model generalizes well to unseen data, as what we have expected.