Lab 6-Reflections on transfer learning
By: Abhash Shrestha (35464984) as14n23@soton.ac.uk

1. Finetuning

The process of fine-tuning involves training the model with our own data. As the network is already largely trained, we use a small learning rate so that we do not make big changes in weights.

Also, freezing the layers preserves the pre-trained weights, which have already learned useful features from the ImageNet dataset, ensuring these features are not destroyed during the initial phases of training on the new dataset.
Allowing the last layer to update its weights gives the model flexibility to adjust its outputs according to the features of the new dataset

The results for the first model can be see in fig 1.

I used the Cross Entropy Loss Function and the Adam optimizer.

Cross Entropy emphasizes improving confidence in the correct classification by penalizing incorrect classifications based on the distance from the correct answer in probability terms. It is a standard choice in classification scenarios as it provides a probability distribution over multiple classes.

Adam has an adaptive learning rate capabilities, and is particularly suitable for fine-tuning because it can make small adjustments to the weights. This is suitable for this task as we only need to make subtle changes.

	precision	recall	f1-score	support
Alilaguna	0.50	0.58	0.54	19
Ambulanza	0.10	0.09	0.09	22
Barchino	0.00	0.00	0.00	51
Gondola	0.00	0.00	0.00	3
Lanciafino10m	0.00	0.00	0.00	7
Motobarca	0.00	0.00	0.00	59
Motopontonerettangolare	0.00	0.00	0.00	3
MotoscafoACTV	0.00	0.00	0.00	1
Mototopo	0.60	0.91	0.72	274
Patanella	0.31	0.49	0.38	74
Polizia	0.00	0.00	0.00	15
Raccoltarifiuti	0.00	0.00	0.00	19
Sandoloaremi	0.00	0.00	0.00	3
Topa	0.00	0.00	0.00	29
VaporettoACTV	0.98	0.97	0.97	325
Water	0.97	0.95	0.96	420
accuracy			0.77	1324
macro avg	0.22	0.25	0.23	1324
weighted avg	0.70	0.77	0.72	1324

fig 1: fine tuned resnet50

2. Reflect on the two different approaches

The SVC had the following results: fig 2 .
These results are much better than the last one where we were using resnet50.

The SCV was used with the following hyper-parameters:

def __init__(*, C=1.0, **kernel='rbf'**, degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None)

The RBF kernel helps in improving the performance, which we can see above.
This is because the RBF kernel is better at handling of margin optimization and non-linear decision boundaries.

The runtime was also significantly reduced. While running on Google Collab, the first model tool 15 minutes to run where as the second was completed within 18 seconds.

The computational overhead of deep learning and a complex model is also reduced since only the feature extractor part needs to be trained.
SVMs, especially with a fixed feature set, is quicker to train compared to optimizing the weights of a deep neural network's fully connected layers. This is reflected in the reduced training time of the second model.

	precision	recall	f1-score	support
Alilaguna	0.90	1.00	0.95	19
Ambulanza	0.82	0.82	0.82	22
Barchino	0.76	0.25	0.38	51
Gondola	1.00	0.67	0.80	3
Lanciafino10m	0.00	0.00	0.00	7
Motobarca	0.82	0.31	0.44	59
topontonerettangolare	1.00	1.00	1.00	3
MotoscafoACTV	0.00	0.00	0.00	1
Mototopo	0.82	0.99	0.89	274
Patanella	0.41	0.84	0.55	74
Polizia	0.67	0.13	0.22	15
Raccoltarifiuti	1.00	0.74	0.85	19
Sandoloaremi	0.00	0.00	0.00	3
Topa	0.00	0.00	0.00	29
VaporettoACTV	0.99	1.00	1.00	325
Water	0.99	0.97	0.98	420
accuracy			0.87	1324
macro avg	0.64	0.54	0.56	1324
weighted avg	0.87	0.87	0.85	1324

fig 2: metrics for model 2 (SVC)