Transfer Learning Report

George Andrianopoulos

August 11, 2022

Final scoring results

Model	Fixed features test score	Fine tune test score
MobileNet_V2	0.82	0.91

The conda.yml specifies the conda environment needed for the execution. The main script for submitting a job is train_SBATCH_keras.sh, and report_SBATCH_keras.sh can be used to generate the nice figures. Those are scripts for SLRUM technology.

Fixed feature classifier training

I split the hyper-parameter search for the fixed feature classifier by the transfer learning model. Each research searched for the 'best' hyper-parameters manually. The principle hyper-parameters that I used were the number and kind of layers laid on top of the model

I was focused on hyperparameter tuning on the number of the unfrozen layers and the order, activation function, learning rate, batch size, number of epochs, adding a flatten layer instead of a pooling layers, adding 2 convolutional layers before the dense layers and data augmentation methods.

In appendix 1 there are some of our architectures that were unsuccessful.

Best models:

- MobileNet: **Dense (256) – Dense (128),** activation = relu, optimizer (fixed feature) = Adam (learning rate = 0.001) , batch size = 16, data augmentation = random_flip, MixUp, epochs = 80, globalpooling
- In appendix 2 I provide the architecture for best model.

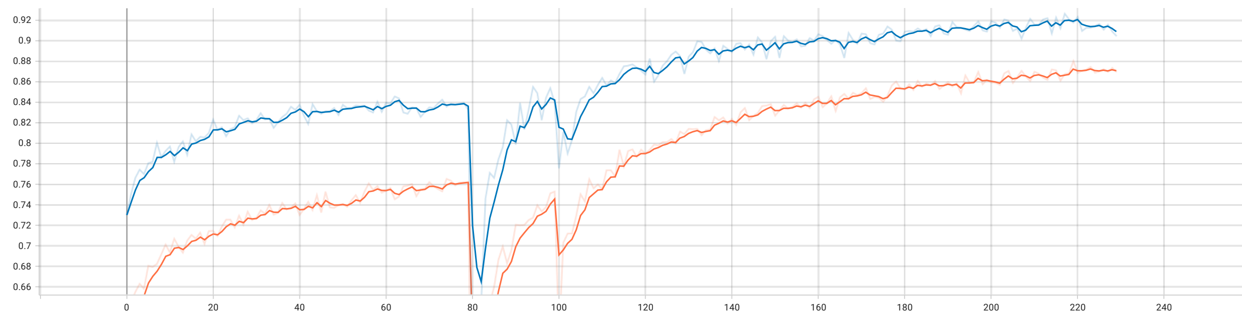
Fine-tuning

I adjusted the number of layers that I unfroze and ended up with 70 unfrozen layers. However, I did not seem to run to an over-fitting problem. I observe overfitting during the initial test where the data weren't augmented. In MobileNet overfitting started after the 220 where I stop the training procedure.

I achieved a very well performing model during fine-tuning, as shown in figure 1. I experimented with the number of unfrozen layers and settled on 70. By the end of training the frozen model, the validation accuracy was around 0.76 at epoch 80. By the end of fine-tuned training in epoch 119, the validation accuracy was 0.92.

Figure 1: MobileNet accuracy

Training (orange) and validation (blue) accuracy over 80 epochs of fixed-feature training and another 150 epochs of fine-tuned training. Best model on epoch 219



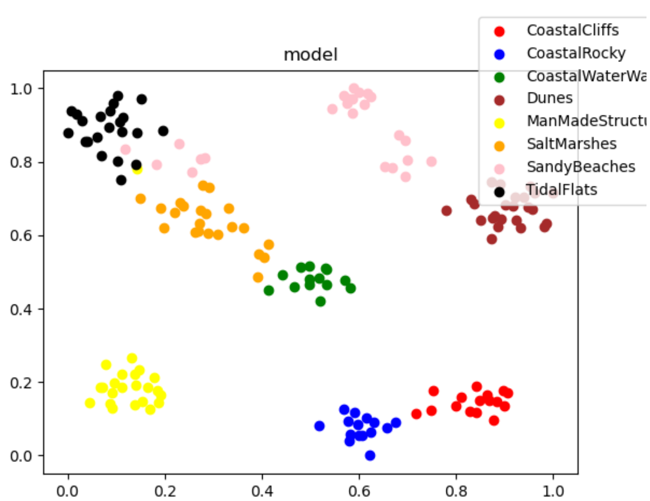
The confusion matrix shows that the model distinguished between the first five categories but struggled with the sandy beaches and tidal flats. In appendix 3 and 4 I present the evaluation plots and classification reports respectively. During the training procedure I use those plots in order to get a better understanding of how my model performs and what hyperparameters I should change.

Another hyperparameter that was highly correlated with the final performance was the change of the optimizer during the fine tuning. SGD with a lower learning rate performed better.

Cluster model embeddings

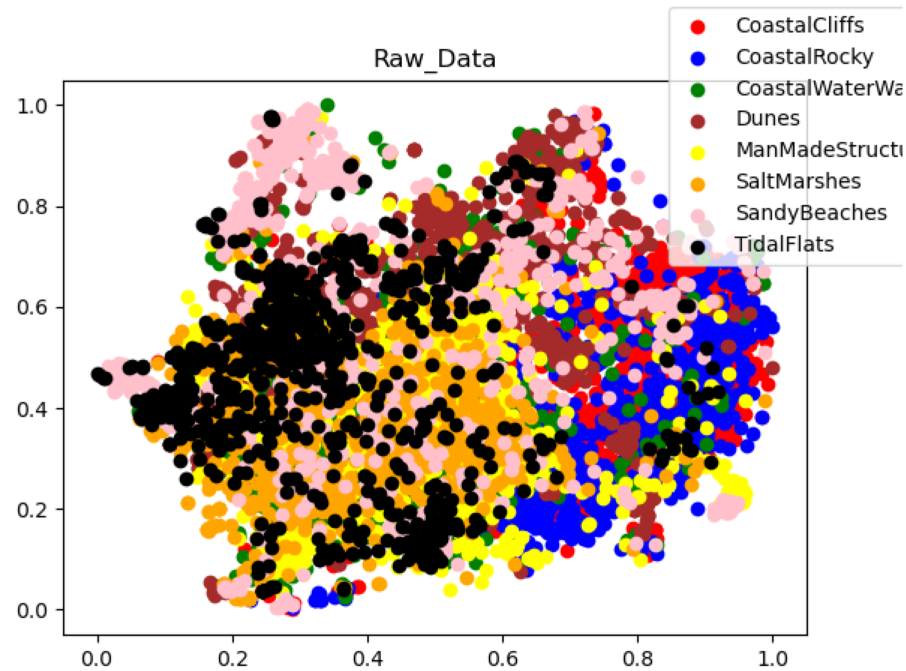
A problem that I faced was that during the extraction of the components I got an error that I ran out of memory. To solve it I create batches of 5 features sets and then extract the components. According to the tSNE visualizations and even if we know that is not an accurate visualization. because of the use of only 2 strongest components, all our models struggled with the same coastlines. For example, in figure 2, MobileNet fine-tuned classifier grouped some 'sandy beaches' together, but also clustered 'sandy beaches' near the 'tidal flats' and the 'dunes'.

Figure 2: tSNE clustering for fine-tuned MobileNet.



The models made a significant improvement to the raw data as can be seen in figure 3.

Figure 3: tSNE clusters for the raw dataset. What a difference!



APPENDIX 1

Table 1: Unsuccessful models with their architecture and evaluation

Model No	data augmentation	learning rate	batch size	n_epochs	fine tuning epochs	added_layers	activation function	unfrozen layers	accuracy fixed features	loss fixed features	accuracy fine tunning	loss fixed features
1	random_flip	0.005	32	60	60	dense (256)	sigmoid	100	0.8312	0.5987	0.8187	0.6018
	MixUp											
	random_augment											
2	random_flip	0.001	32	60	40	flatten (128000)	sigmoid	90	0.4775	1.4319	0.68	1.22
	MixUp					dense (400)						
	random_augment					dropout (400)						
3	random_flip	0.001	16	100	40	dense (256)	sigmoid	90	0.6812	1.1632	0.7	1.1039
	MixUp					dropout (256)						
						dense (64)						
4	random_flip	0.001	16	100	100	dense (256)	sigmoid	90	0.3692	1.78	0.42	1.5938
	MixUp					dense (64)						
5	random_flip	0.01	64	100	100	dense (256)	relu	90	0.125	2.3858	0.125	2.0798
	MixUp					dropou (256)						
						dense (64)						
6	random_flip	0.001	16	100	100	dense (256)	relu	90	0.54	1.28	0.5512	1.2516
	MixUp					dense (64)						
7	random_flip	0.001	16	100	100	conv2d (8, 8, 32)	relu	90	0.6338	1.0952	0.6463	1.05105
	MixUp					max_pooling2d (4, 4, 32)						
						flatten (512)						
						dense (256)						
						dense (128)						

APPENDIX 2

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 299, 299, 3)]     0

 tf.math.truediv (TFOpLambda  (None, 299, 299, 3)      0
 )

 tf.math.subtract (TFOpLambd  (None, 299, 299, 3)      0
 a)

 mobilenetv2_1.00_224 (Funct  (None, 10, 10, 1280)     2257984
 ional)

 global_average_pooling2d (G  (None, 1280)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 256)               327936

 dense_1 (Dense)             (None, 128)               32896

 dense_2 (Dense)             (None, 8)                 1032

=================================================================
Total params: 2,619,848
Trainable params: 769,768
Non-trainable params: 1,850,080
_________________________________________________________________
```

Figure 4: Architecture of MobileNet

APPENDIX 3

MobileNet

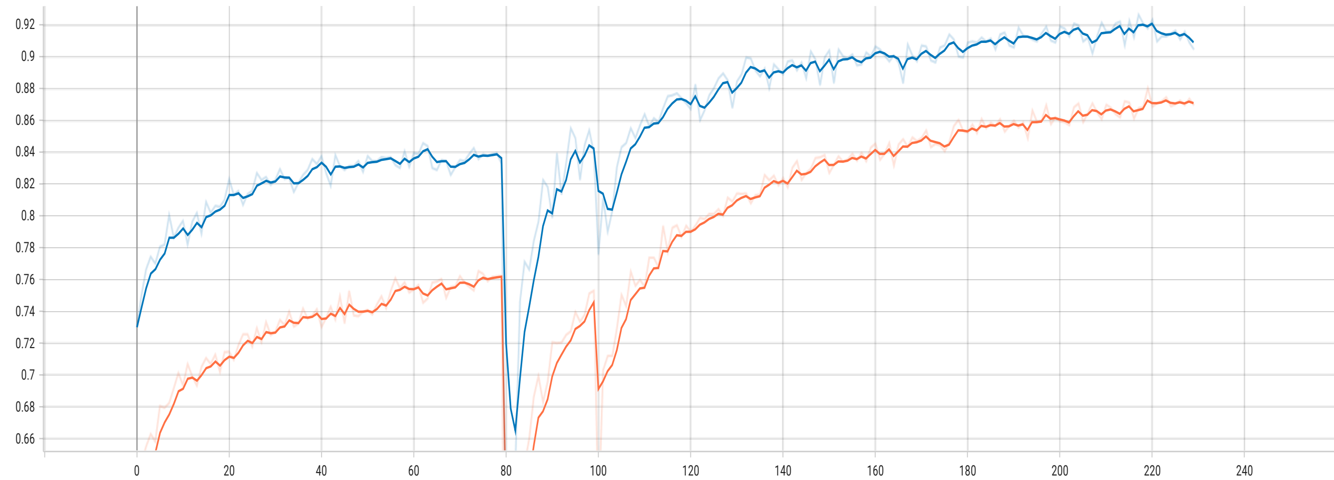


Figure 5:epoch accuracy. validation acc (blue) training acc (orange)

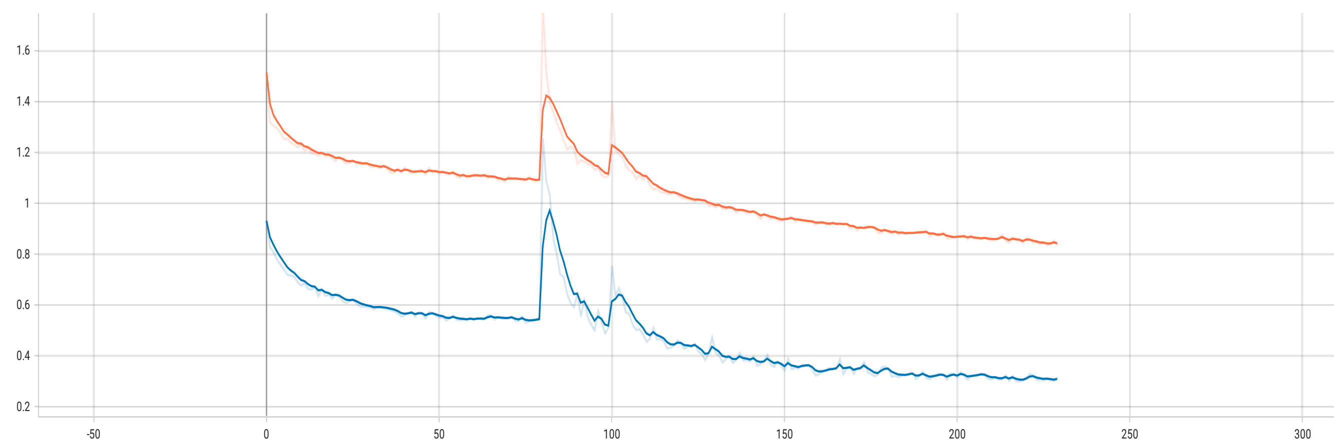
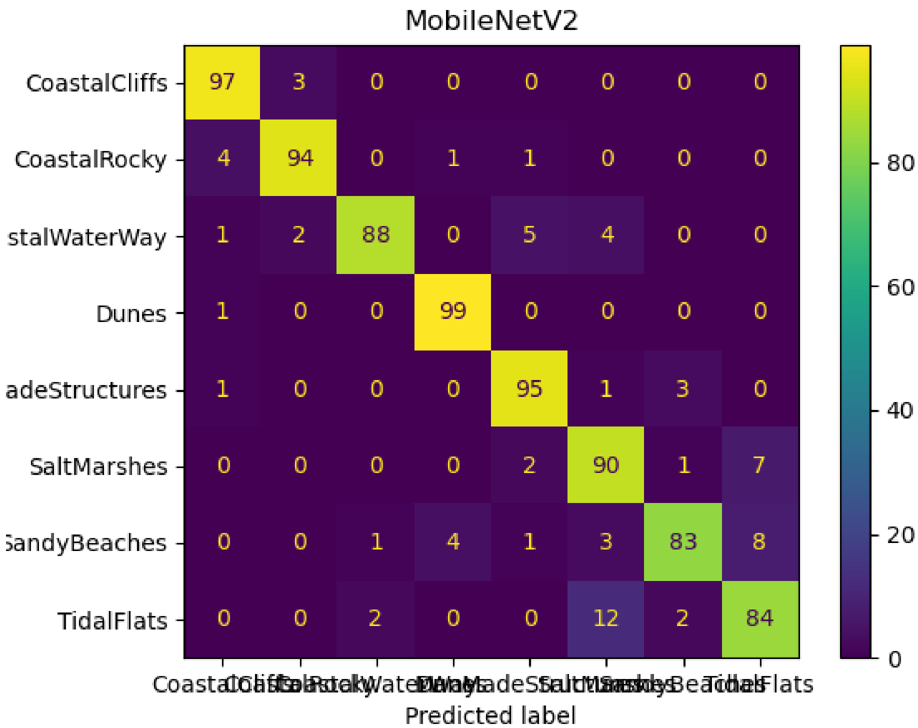


Figure 6: epoch loss. validation loss (blue) training loss (orange)

APPENDIX 4

MobileNet



confusion matrix

	precision	recall	f1-score	support
0	0.93	0.97	0.95	100
1	0.95	0.94	0.94	100
2	0.97	0.88	0.92	100
3	0.95	0.99	0.97	100
4	0.91	0.95	0.93	100
5	0.82	0.90	0.86	100
6	0.93	0.83	0.88	100
7	0.85	0.84	0.84	100
accuracy			0.91	800
macro avg	0.91	0.91	0.91	800
weighted avg	0.91	0.91	0.91	800

classification report