Assignment: Transfer Learning on Intel Image Classification

David Bertoldi – 735213
email: d.bertoldi@campus.unimib.it

Department of Informatics, Systems and Communication

University of Milano-Bicocca

1 Dataset

The chosen dataset is called Intel® Image Classification and it was initially published on Analytics Vidhya by Intel® to host an image classification challenge to promote OpenVINO™, a toolkit or optimizing and deploying AI inference [1][2].

The dataset contains images of natural scenes around the world and they belong to 6 classes: buildings, forests, glaciers, mountains, sea and streets. The images are of size 150 × 150px and can be colored (3 channels, RGB) or rarely in grayscale (still with 3 channels). Figure 1 shows 16 entries of the training dataset.

There is a total of $\sim 24\,000$ images, divided into Train ($\sim 14\,000$), Test ($\sim 3\,000$) and Prediction ($\sim 7\,000$) folders. The last one does not contain labels and it is intended for unsupervised learning and it will be ignored in this work.

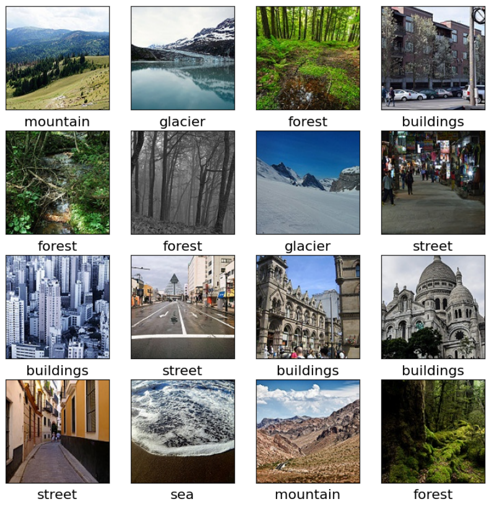


Figure 1: 16 random entries of the train dataset

The distribution of the images across the classes follows a uniform distribution $U(\mu, \sigma)$: in the train set each class has an average $\mu = 2\,339$ images with $\sigma = 105.45$ and in the test set $\mu = 500$ and $\sigma = 36.92$. We didn't find any bias inside the dataset since all the classes are equally populated and so we didn't applied any kind of data augmentation on particular classes for rebalacing.

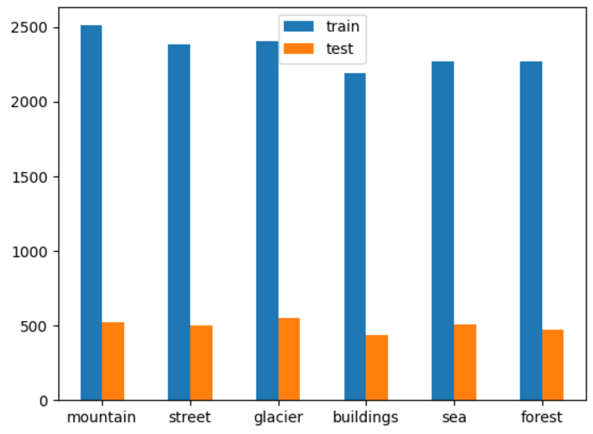


Figure 2: 16 entries of the train dataset

The 6 classes are encoded with numbers 0 to 5 and Table 1 shows the mapping between the numerical and nominative form.

Number	Class
0	Mountain
1	Street
2	Glacier
3	Building
4	Sea
5	Forest

Table 1: Mapping between numbers and names

2 The model

The chosen dataset presented similarieties with ImageNet: the 6 classes of Intel® Image Classification are scattered and distributed in the $1\,000$ classes of ImageNet. For this reason a pretrained model on ImageNet speeded up the learning process. The chosen model is VGG16, a 16−layers deep CNN proposed by Karen Simonyan and Andrew Zisserman at the

Cut	Trainable parameters	Dimension
`fc1`	117 479 232	$1 \times 1 \times 4096 = \mathbf{4\,096}$
`block4_pool`	7 635 264	$14 \times 14 \times 512 = \mathbf{100\,352}$
`block3_pool`	1 735 488	$28 \times 28 \times 256 = \mathbf{200\,704}$

Table 2: Number of VGG16's trainable parameters and dimensions of the extracted features at each cutting point

University of Oxford[3]. Figure 3 shows an overview of its architecture.

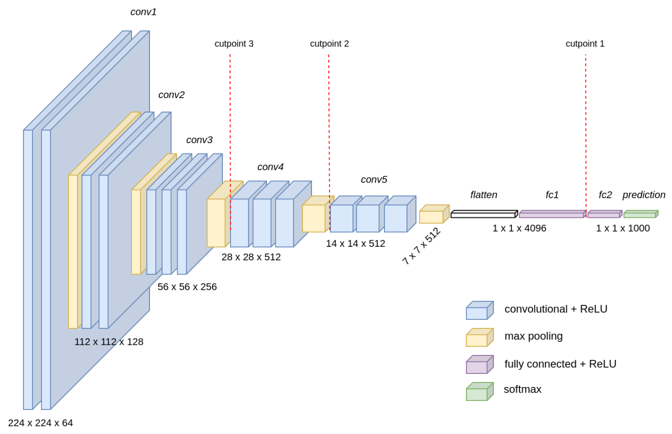


Figure 3: Architecture of VGG16 with the cuts applied in this work

In this work we proposed different cuts to the network and feeded its outputs to a ”classic” machine learning model, a SVM, and benchmark the performance of the hybrid architecture.

The chosen cuts were after the first dense layer (`fc1`), after the fourth pooling layer (`block4_pool`) and after the third pooling layer (`block3_pool`). The different cuts led to different challenges, such as the high dimensionality of the features.

As the cuts approached the input, the number of trainable parameters decreased exponentially; however the representation of the features will have an increasingly higher dimension. The higher dimensionality affects the training performance of the SVM and a fine tuning on the management of the memory. For this reason we used less samples during the training phase as the dimensionality increased, raising the risk of underfitting. Table 2 describes the numbers of the problems.

3 First cutting point: `fc1`

Before attaching to the layer `fc1` the SVM, it is interesting to visualize the features through PCA (Principal Component Analysis): we decreased the number of dimensions of the extracted features from 4 096 to 2 and plotted the graph in Figure 4.

In the projection through PCA are identifiable some clusters: glaciers and mountains are very similar, the same goes for streets and buildings. This make sense since the picture of the mountains may contain snow or ice and picture of buildings may contain the surrounding street, and viceversa. The seas borders with the glaciers but it does not blend with them. The forests are a separate concepts and it seems to not share anything with the other classes.

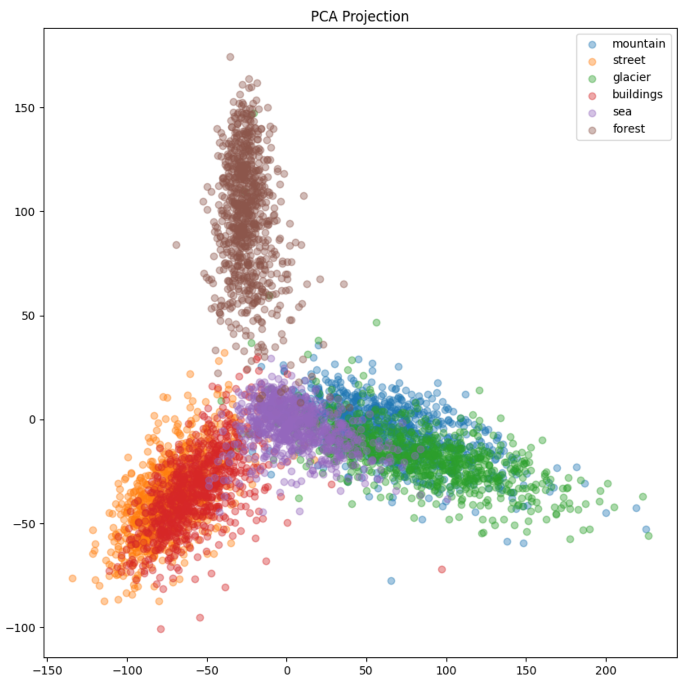


Figure 4: Architecture of VGG16 with the cuts applied in this work

Because the cut is very close to the VGG16's original output we expected an high accuracy in classifying the images.

3.1 Extraction of the features and training of the SVM

Before starting with the training of the hybrid model, we chose the SVM's hyperparameters using `GridSearchCV` from sklearn. This function does an exhaustive search of the best parameters for the SVM in order to find the best separation hyperplane. The parameters to be chosen were the cost $C \in \{5, 10, 20, 100\}$ and the kernel coefficient $\gamma \in \{scale, auto\}$. This does not assure that it gives back the best configurations on unseen data. As a matter of fact we should had the best results with cost $C = 100$ and $\gamma = auto$ (Figure 5), but in this case we achieved higher performances on the test dataset with $C = 100$ and $\gamma = scale$. So we used `GridSearchCV` just like a starting point for further manual experimentations.

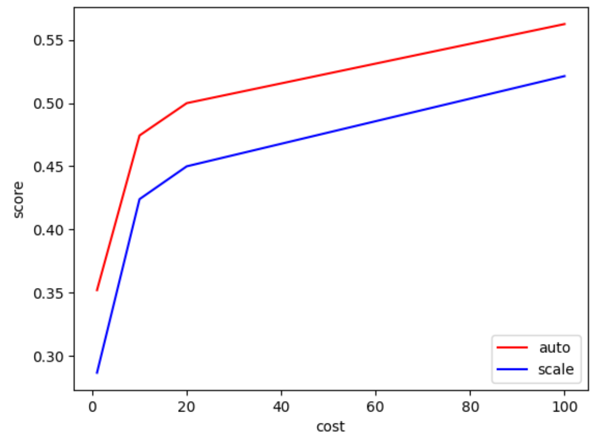


Figure 5: Model's performance on the training set with different configurations

An issue when using hybrid models are some incompatibilities between libraries: we started using Tensorflow's

`DirectoryIterator` or `Dataset` in order to efficiently load the dataset in batches and to not saturate the memory; unfortunately sklearn does not support this kind of data structure and the only feasible solution was to diminuish the number of samples during the training and test phases. With this cutting point we used 1 000 images per class when training and 450 when testing.

The training accuracy reached 100% and Figure 6 shows the confusion matrix.

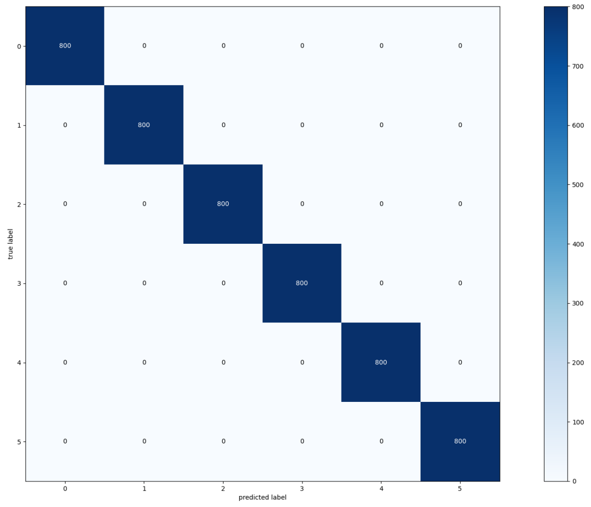


Figure 6: Confusion matrix on training dataset (cut at `fc1` + SVM($C = 100, \gamma = scale$))

The test accuracy reached 92% and Figure 7 shows the confusion matrix. We can notice a little confusion between mountains and glaciers and between streets and buildings. This results are in accordance with Figure 4.

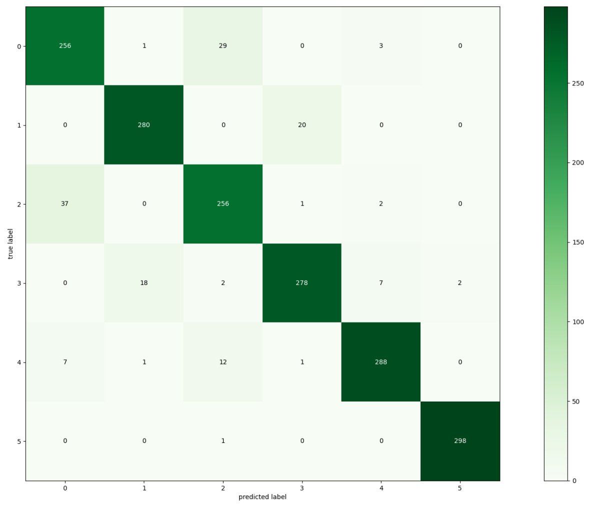


Figure 7: Confusion matrix on test dataset (cut at `fc1` + SVM($C = 100, \gamma = scale$)))

The training phase for SVM took ~ 550 seconds. We tried to simplify the model and thus decrease the timings by reducing the dimensionality of the features. In order to do so we used PCA as technique for dimensionality reduction.

3.2 Dimensionality Reduction

In this section we describe how we achieved high accuracy by reducing the dimensionality of the features in input.

Before applying PCA, we estimated how many components were needed to describe the data. This can be determined by looking at the *cumulative explained variance* ratio as a function of the number of components.

The curve in Figure 8 quantifies how much of the total 4096−dimensional variance is contained within the first n components. For example, we saw that the first ~ 400 components contain approximately 80% of the variance, while are needed around 2 500 components to describe 100% of the variance. We chose a configuration that could preserve 90%

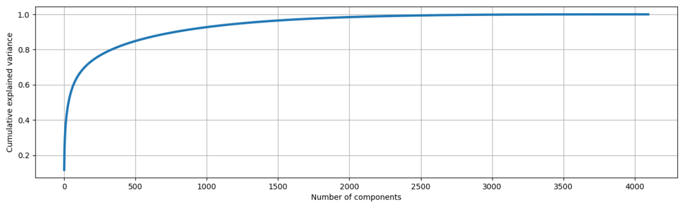


Figure 8: Confusion matrix on test dataset (cut at `fc1` + SVM($C = 100, \gamma = scale$)))

of the variance, *i.e.* 777 components. This could assure good perfomances while using just 19% of the dimensions.

In this case we found the optimal configuration of the SVM with $C = 5$ and $\gamma = scale$.

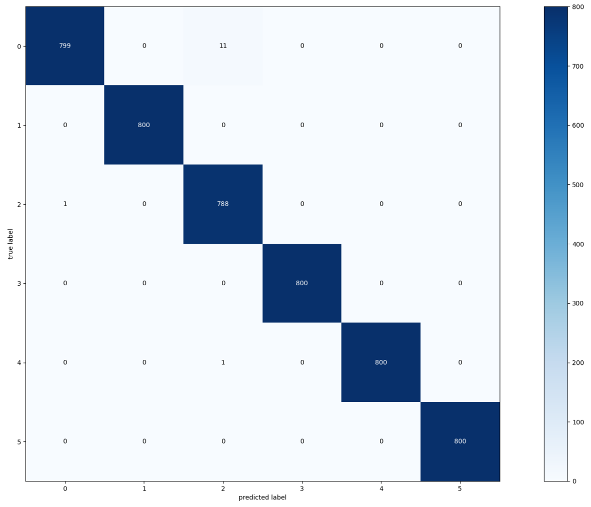


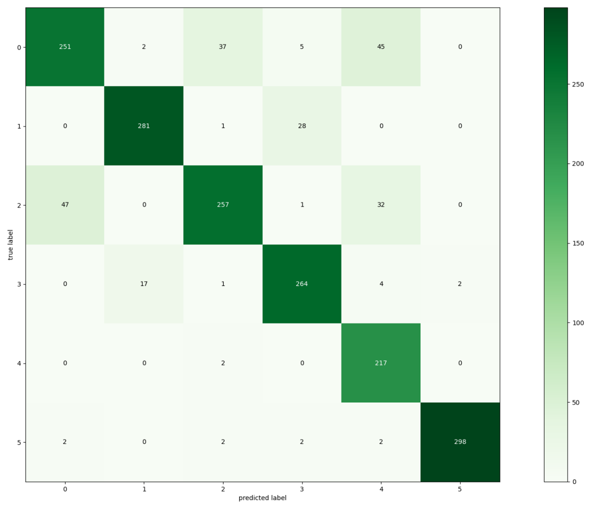
Figure 9: Confusion matrix on training dataset (cut at `fc1` + PCA + SVM($C = 5, \gamma = scale$))

Figure 9 and 10 shows the confusion matrix after training and testing. We can see that there is a slight worsening in training accuracy (99.9%) while test accuracy dropped to 88%. We noticed that the model still confused glaciers with mountain and streets with buildings; in addition to them, some of the images of the sea are mistaken for mountains and glaciers. That means the model is affected by more overfitting before applying PCA.

The lower accuracy and higher overfitting are traded off with faster times in training: we measured a speed up of almost 10 times, reaching 57 seconds.

4 Second cutting point: `block4_pool`

Like in section 3 we repeated the experiment of visualizing the features extracted from layer `block4_pool` through PCA and plotted the graph in Figure 11.



	0	1	2	3	4	5
0	251	2	37	5	45	0
1	0	281	1	28	0	0
2	47	0	257	1	32	0
3	0	17	1	264	4	2
4	0	0	2	0	217	0
5	2	0	2	2	2	298

Figure 10: Confusion matrix on test dataset (cut at `fc1` + PCA + SVM($C = 5, \gamma = scale$))

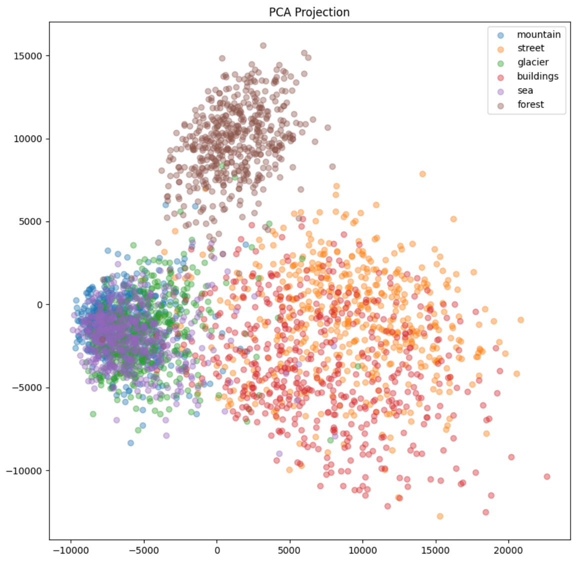


Figure 11: Architecture of VGG16 with the cuts applied in this work

This time the sea, glacier and mountain classes collapsed in one cluster. This anticipated that images belonging to these classes could be classified with lower accuracy. The cluster of images of forest is the only one that did not collide with any other. We also noticed more isolated images being completely misclassified than the previous cut, *e.g.* some images of mountain or sea are predicted as building or street.

References

[1] Practice Problem: Intel Scene Classification Challenge
`https://datahack.analyticsvidhya.com/contest/practice-problem-intel-scene-classification-challe`

[2] OpenVINO™ documentation
`https://docs.openvino.ai/latest/index.html`

[3] *Very Deep Convolutional Networks for Large-Scale Image Recognition*
Karen Simonyan, Andrew Zisserman
`https://doi.org/10.48550/arXiv.1409.1556`