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 for optimizing and deploying AI inferences [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 are colored (3 channels, RGB) or more 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.

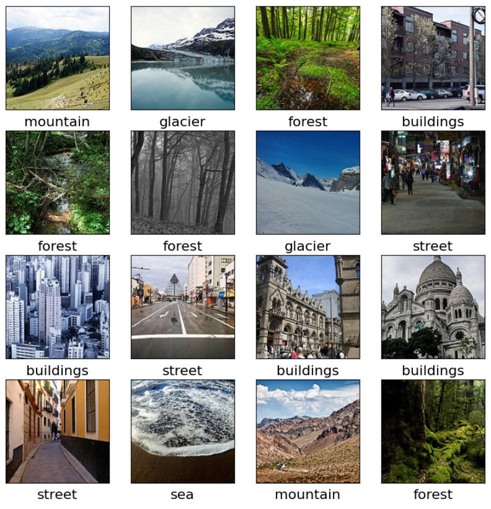


Figure 1: 16 random entries of the train dataset

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.

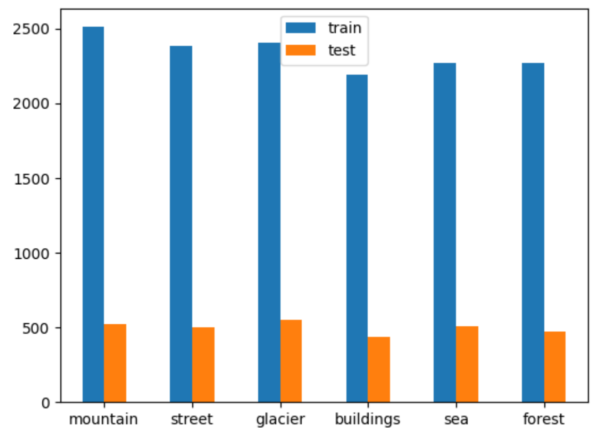


Figure 2: 16 entries of the train dataset

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 similarities 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 sped up the learning process. The chosen model is VGG16, a 16−layers deep CNN proposed

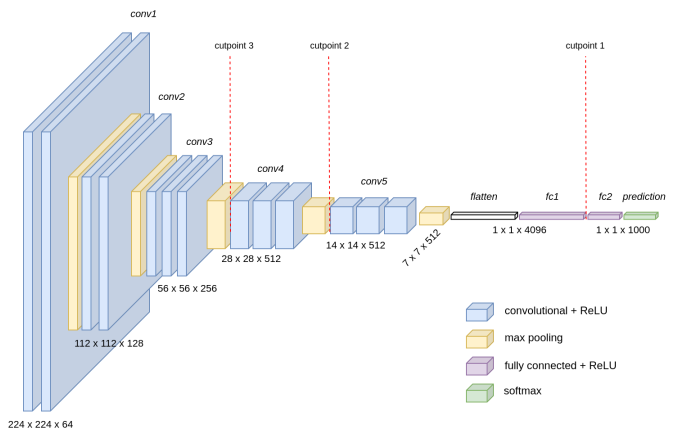


Figure 3: Architecture of VGG16 with the cuts applied in this work

by Karen Simonyan and Andrew Zisserman at the University of Oxford[3]. Figure 3 shows an overview of its architecture.

In this work we proposed different cuts to the network and fed the extracted features into a ”classic” machine learning model, a SVM, and made a benchmark of the performances of this 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 very high dimensionality of the features.

As the cuts approached the input, the number of trainable parameters decreased exponentially; however the representation of the features had an increasingly higher dimension. The higher dimensionality affected the training performance of the SVM and a fine tuning on the management of the memory was required. For this reason we used less samples during the training phase as the dimensionality increased, raising the risk of underfitting. Table 2 shows the details at each stage.

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

3 Preprocessing

Each image was preprocessed in the same way images from ImageNet were preprocessed during the training of VGG16. That is each image is converted from RGB to BGR, then each color channel is zero-centered with respect to the ImageNet dataset, without scaling. This guaranteed that the performances VGG16 demonstrated with ImageNet could be reproduced with this dataset.

4 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.

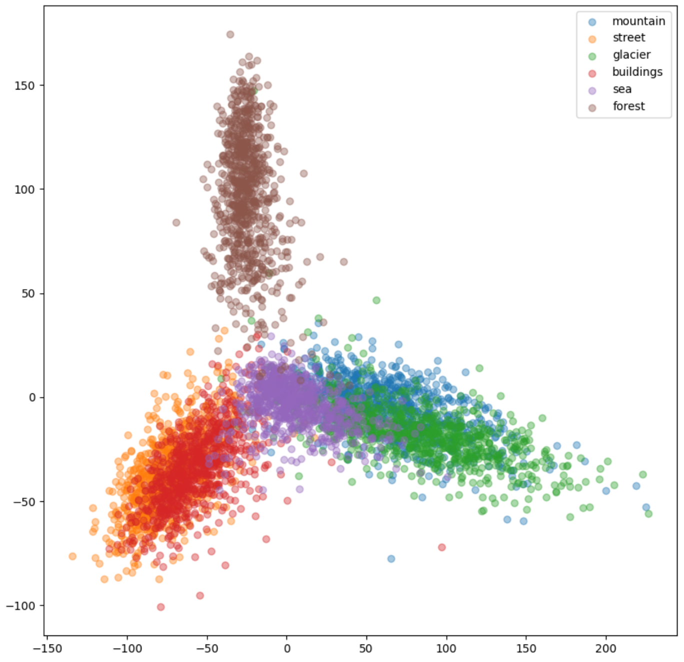


Figure 4: PCA projection at `fc1` layer

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 vice versa. 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.

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

4.1 Extraction of the features and training of the SVM

Before starting with the training of the classic model, we chose the SVM’s hyper parameters 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 experimentation.

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 diminish the number of samples during the training and test phases. With this cutting point we used 800 images per class when training and 300 when testing.

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

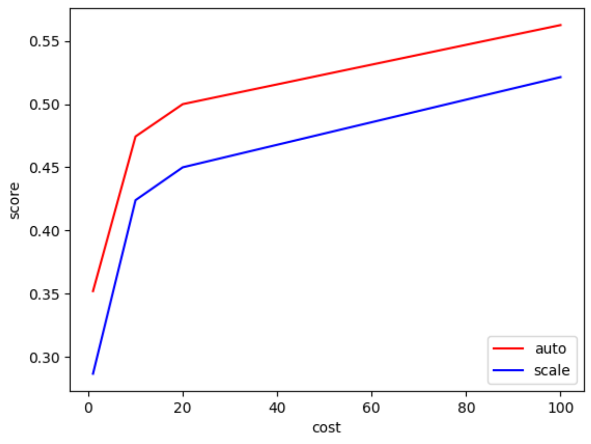


Figure 5: Model's performance on the training set with different configurations

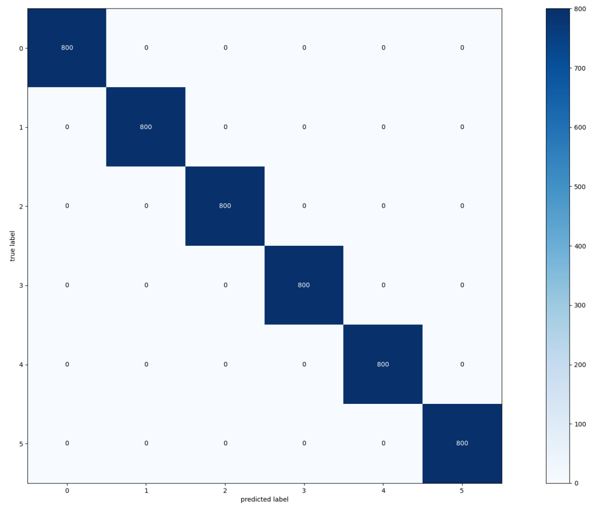


Figure 6: Confusion matrix on training dataset (cut at `fc1` + SVM($C = 100, \gamma = scale$))

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.

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.

4.2 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 almost 100% of the variance. We chose a configuration that could preserve

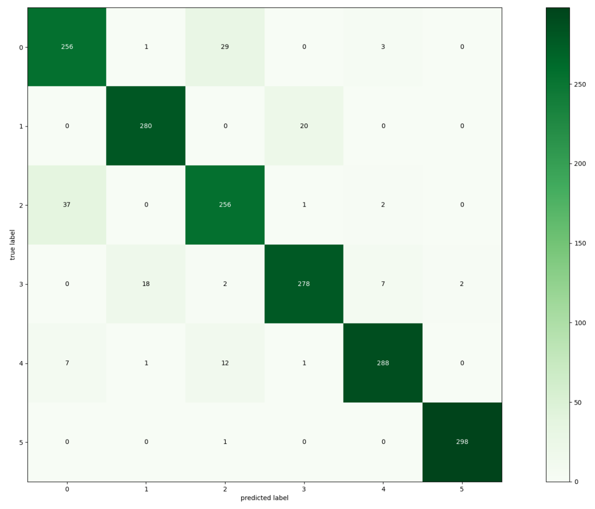


Figure 7: Confusion matrix on test dataset (cut at `fc1` + SVM($C = 100, \gamma = scale$)))

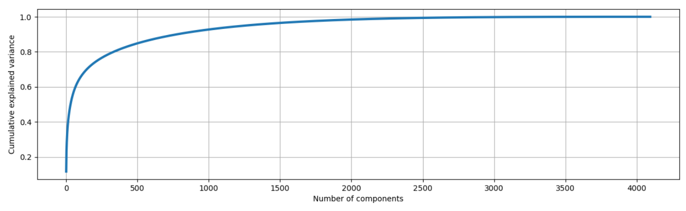


Figure 8: The cumulative explained variance of the 4 096 components

90% of the variance, *i.e.* 777 components. This could assure good performances while using just 19% of the dimensions.

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

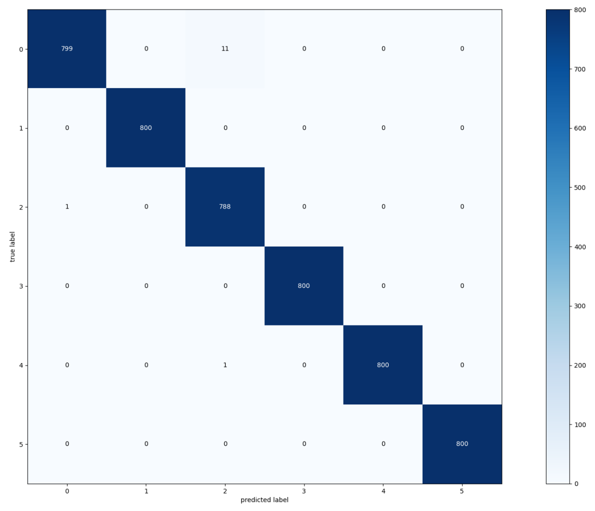


Figure 9: Confusion matrix on training dataset (cut at `fc1` + PCA + SVM($C = 5, \gamma = scale$))

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

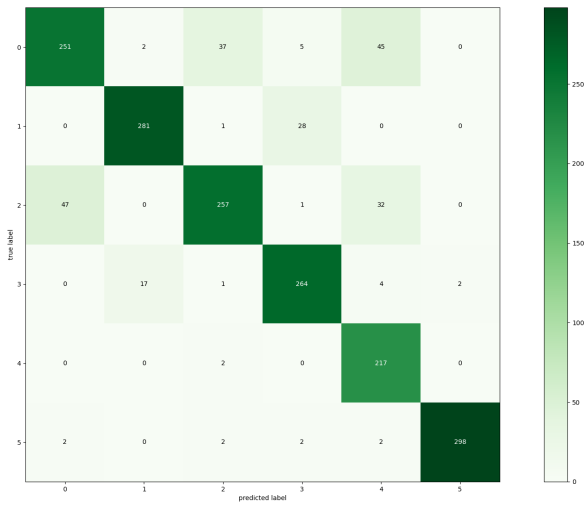


Figure 10: Confusion matrix on test dataset (cut at `fc1` + PCA + SVM($C = 5, \gamma = \mathit{scale}$))

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.

5 Second cutting point: `block4_pool`

Like in section 4 we repeated the experiment of visualizing the features extracted from layer `block4_pool` through PCA and plotted the graph in Figure 11.

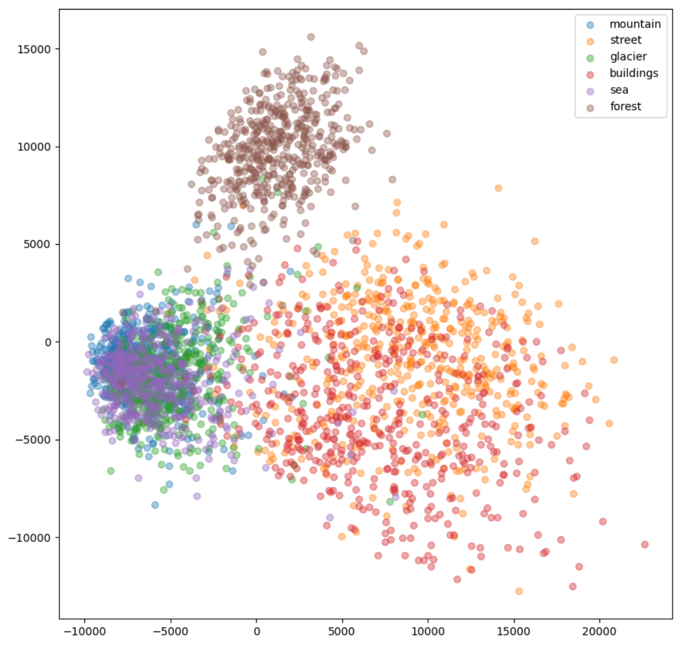


Figure 11: PCA projection at `block4_pool` layer

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.

At this stage the number of trainable parameters of the CNN decreased by $\sim 93.5\%$ but the dimensionality of the features increased 25 times, reaching 100 352. This made the application of PCA more crucial.

5.1 Extraction of the features and training of the SVM

We performed an exhaustive search with `GridSearchCV` plus some manual experimentation and we found that a good configuration for the SVM is $C = 10$ and $\gamma = \mathit{scale}$. Because of the increased dimensionality we used less images for training (from 800 to 500) in order to not saturate the memory.

The training accuracy reached again 100% and Figure 12 shows the confusion matrix.

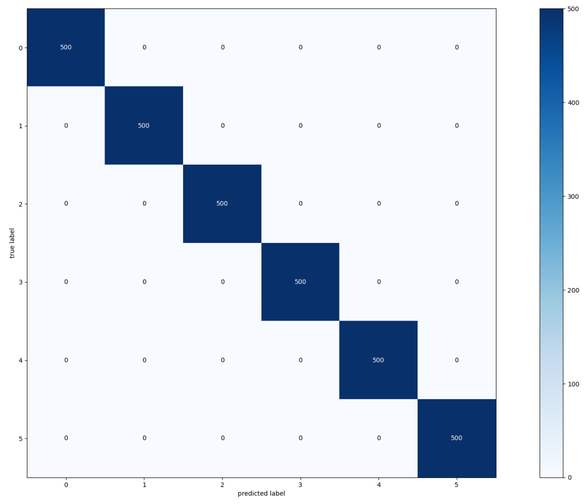


Figure 12: Confusion matrix on training dataset (cut at `block4_pool` + SVM($C = 10, \gamma = \mathit{scale}$))

The test accuracy reached 89% and Figure 13 shows the confusion matrix. We can notice the aforementioned confusion between sea and mountain or glaciers but did not compromise the overall accuracy of the model.

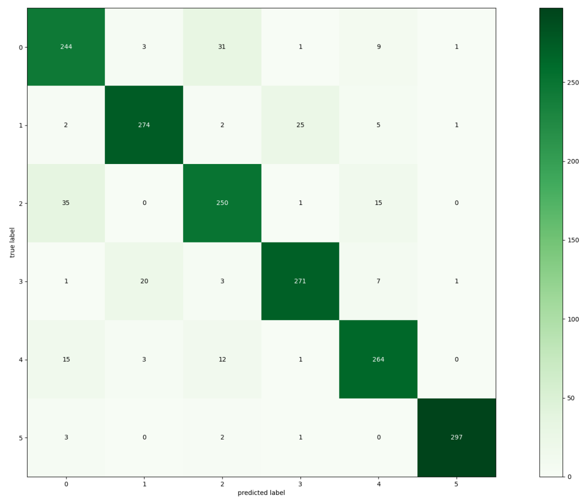


Figure 13: Confusion matrix on test dataset (cut at `block4_pool` + SVM($C = 10, \gamma = \mathit{scale}$)))

5.2 Dimensionality Reduction

We proceeded with the estimation of the optimal number of components in order to describe at least the 90% of the variance. From Figure 14 we can notice that the 90% of the variance can be described with 1 850 components, leading to a compression of 39%.

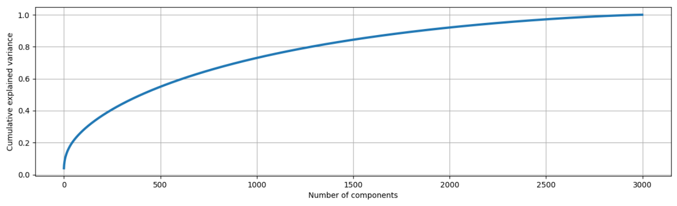
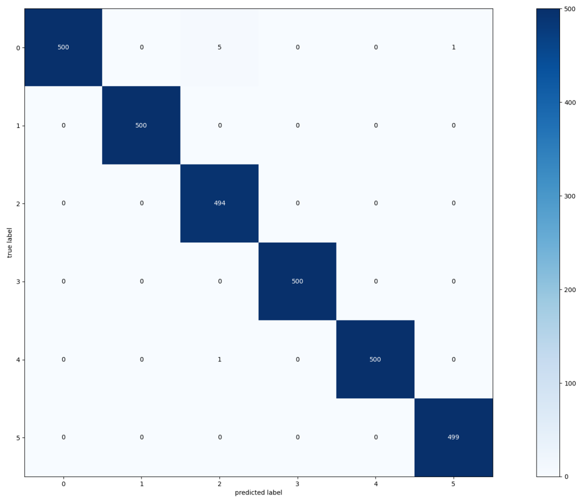


Figure 14: The cumulative explained variance of the 3 000 components

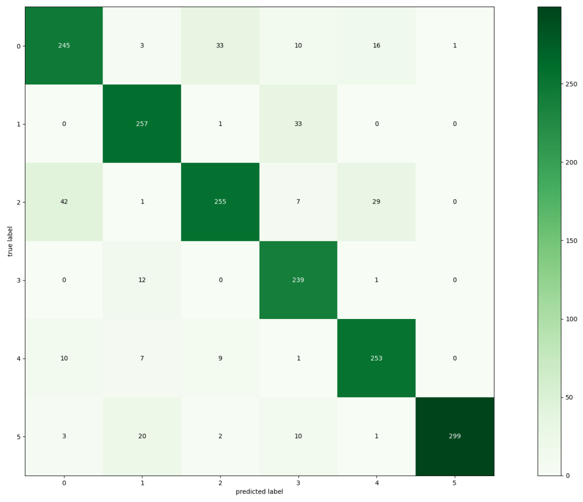
Please note that PCA operated on a dataset $D \in \mathcal{M}_{m\times n}$ calculates the maximum number of components to keep $c = min\{m, n\}$.

In this case the optimal configuration for the SVM was $C = 5$ and $\gamma = scale$; the model reached again 99.9% of accuracy on the training data and 86% on the test data.



true \ predicted	0	1	2	3	4	5
0	500	0	5	0	0	1
1	0	500	0	0	0	0
2	0	0	494	0	0	0
3	0	0	0	500	0	0
4	0	0	1	0	500	0
5	0	0	0	0	0	499

Figure 15: Confusion matrix on training dataset
(cut at `block4_pool` + PCA + SVM($C = 5, \gamma = scale$))



true \ predicted	0	1	2	3	4	5
0	245	3	33	10	16	1
1	0	257	1	33	0	0
2	42	1	255	7	29	0
3	0	12	0	239	1	0
4	10	7	9	1	253	0
5	3	20	2	10	1	299

Figure 16: Confusion matrix on test dataset
(cut at `block4_pool` + PCA + SVM($C = 5, \gamma = scale$))

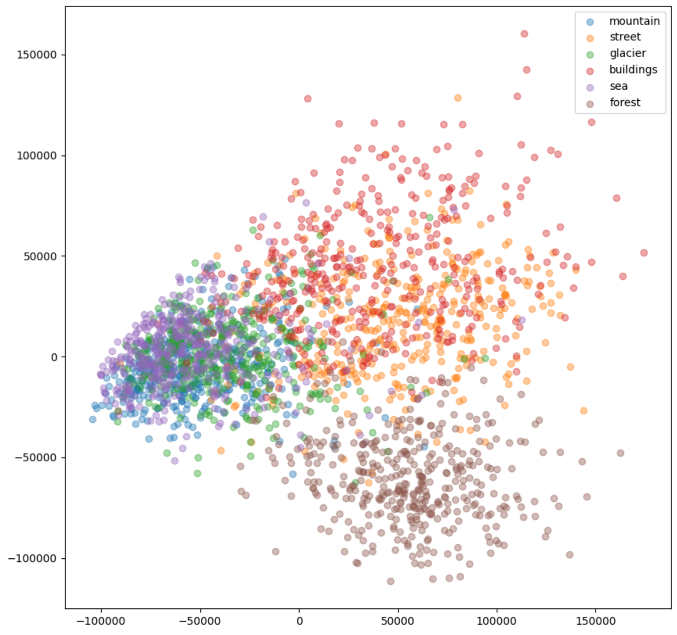


Figure 17: PCA projection at `block3_pool` layer

From Figure 16 we noticed that images of streets or buildings were confused with forests for the first time; the more accurate classifications were registered for images of street, glacier and sea. The application of a technique of dimensionality reduction decreased the test accuracy of a reasonable amount, it didn't resolved the problem of overfitting (which is not guaranteed by PCA) but decreased the training times from 633 seconds to 58 seconds.

6 Third cutting point: `block3_pool`

Like in section 4 we repeated the experiment of visualizing the features extracted from layer `block3_pool` through PCA and plotted the graph in Figure 17.

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.

At this stage the number of trainable parameters of the CNN decreased by $\sim 77\%$ but the dimensionality of the features doubled, reaching 200 704. The application of PCA is even more crucial in this case.

6.1 Extraction of the features and training of the SVM

We performed an exhaustive search with `GridSearchCV` plus some manual experimentation and we found that a good configuration for the SVM is $C = 10$ and $\gamma = scale$. Because of the increased dimensionality we used less images for training (from 400 to 300) in order to not saturate the memory.

The training accuracy reached again 100% and Figure 18 shows the confusion matrix.

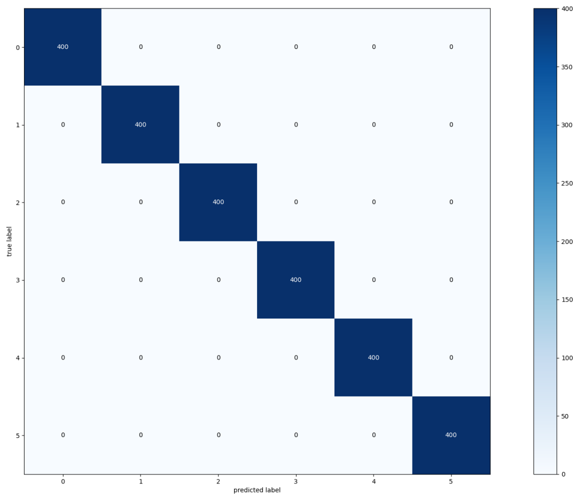


Figure 18: Confusion matrix on training dataset (cut at `block3_pool` + SVM($C = 10, \gamma = \mathit{scale}$))

The test accuracy reached 89% and Figure 19 shows the confusion matrix. We can notice the aforementioned confusion between sea and mountain or glaciers but is not compromising the overall accuracy of the model.

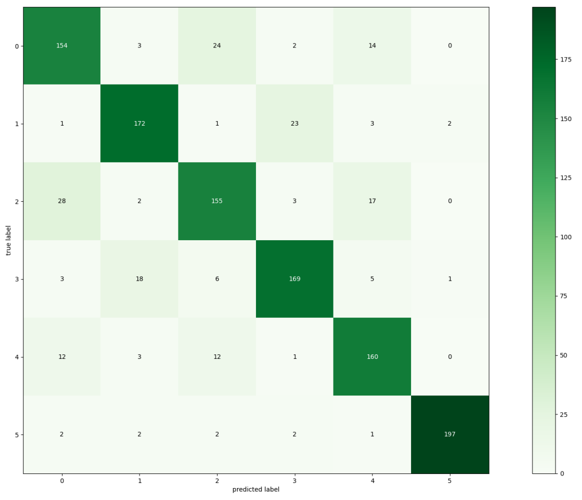


Figure 19: Confusion matrix on test dataset (cut at `block3_pool` + SVM($C = 10, \gamma = \mathit{scale}$)))

6.2 Dimensionality Reduction

We proceeded with the estimation of the optimal number of components in order to describe at least the 90% of the variance. From Figure 20 we can notice that the 90% of the variance can be described with 1 560 components, leading to a compression of 35%.

In this case the optimal configuration for the SVM was $C = 5$ and $\gamma = \mathit{scale}$; the model reached again 99.9% of accuracy on the training data and 77% on the test data.

From Figure 22 we noticed that images of streets or buildings were confused with forests for the first time; the more accurate classifications were registered for images of street, glacier and sea. The application of a technique of dimensionality reduction decreased the test accuracy a lot, it didn't resolved the problem of overfitting (which is not

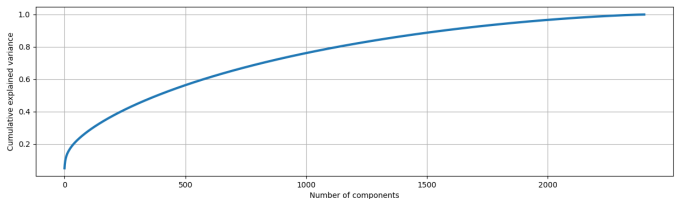


Figure 20: The cumulative explained variance of the 3 000 components

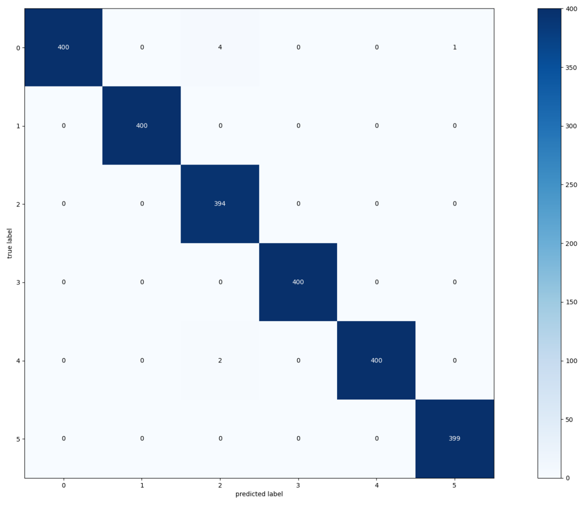


Figure 21: Confusion matrix on training dataset (cut at `block3_pool` + PCA + SVM($C = 5, \gamma = \mathit{scale}$))

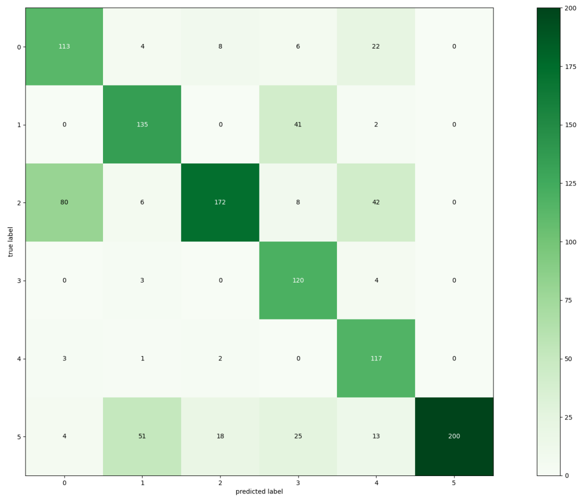


Figure 22: Confusion matrix on test dataset (cut at `block3_pool` + PCA + SVM($C = 5, \gamma = \mathit{scale}$))

guaranteed by PCA) but decreased the training times from 1340 seconds to 160 seconds.

7 Implementation

In this section we describe how to execute the code that helped to produce this work.

Run

```
python3 trainer-1.py --cut fc1 --c1 100 --c2 5
```

to execute the first cut, train the SVM wit $C = 100$, execute the PCA on 90% of explained variance and train the SVM with $C = 5$. Run

```
python3 trainer-1.py --cut block4_pool --c1 10
--c2 5
```

to execute the second experiment.
Run

```
python3 trainer-1.py --cut block3_pool --c1 10
--c2 5
```

to execute the third experiment.

Use `--skip` to skip the first training and jump to PCA. Use `--train <n>` and `--test <n>` to set the number of images for each classes to be used for training and test. Use `--help` for any information about the executable.

8 Conclusions

In general, features extracted at deeper layers (closer to the original output) produced features that guaranteed highly accurate classifications, with 92% and 88% of accuracy on test data before and after PCA. Applying cuttings at higher levels (closer to the input) made the dimensionality of the features increase so much that the usage of techniques of dimensional reduction became crucial in order to use them in the SVM classifier. Performances, in terms of accuracy and training time, got worse and we noted the presence of overfitting especially in the last classifiers.

The results in this works were affected by memory limitations; without these limitations the results may have been different.

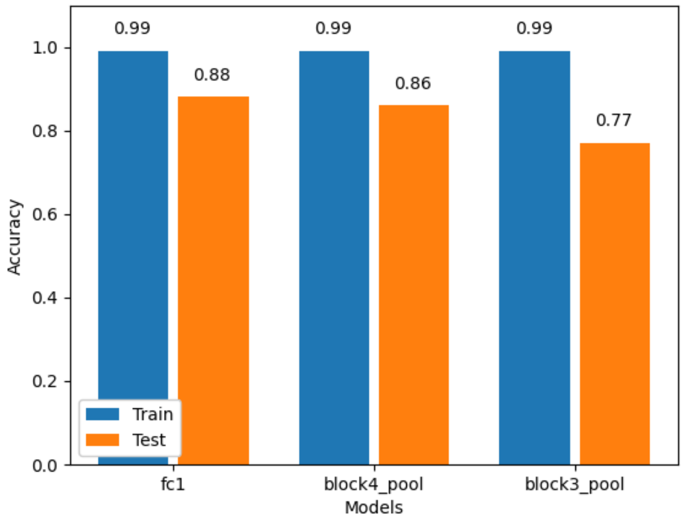


Figure 23: Accuracy after PCA

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`