University of Cagliari

Master degree in Informatics

Supermarket images classification using pretrained networks and linear SVM

Deep Learning Applications Project

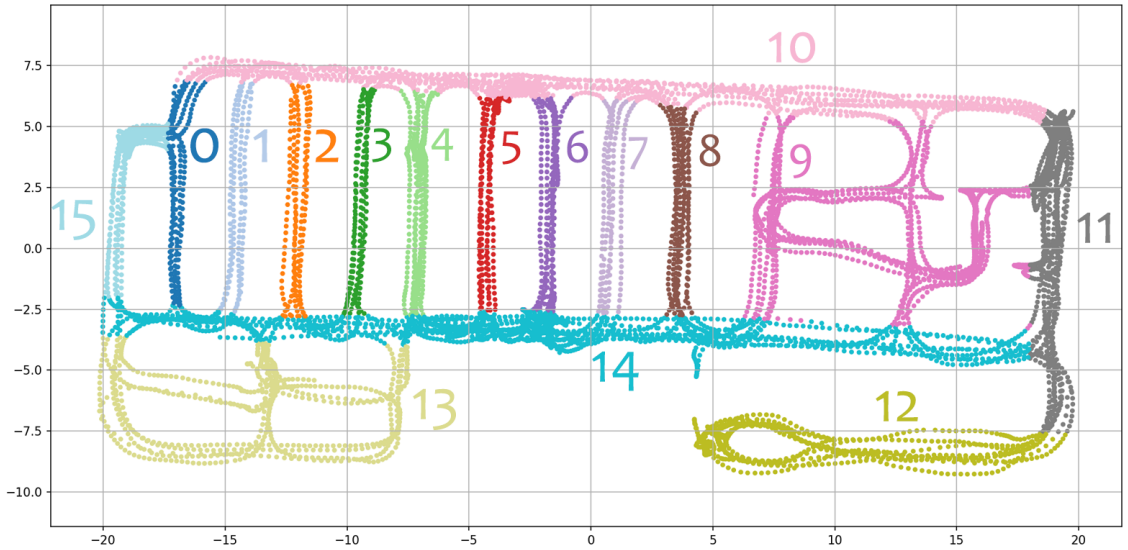
Andrea Piras 65211
Matteo Anedda 65212

Index

Introduction

The objective of this project is to try to understand the position of a person in a supermarket using image classification and deep learning.

The images of the supermarket are taken from a dataset. These images have been taken by a camera attached to a cart that followed some routes inside the supermarket. After the acquisition they have been divided into 16 classes/routes.



With this dataset we will perform the feature extraction using three pretrained networks:

- AlexNet
- ResNet18
- VGG16

After that we will perform the classification using linear SVMs.

Useful Links

- Github project
- Original Dataset
- Organized Dataset

Project Structure

```
.
|
| Folders
├── img   # Images for the readme
├── doc   # Report folder
|    ├── Report.pdf   # Report
├── *TrainingSet*   # Folder with the images of the training set
|    ├── 00
|    ├── 01
|    ├── ...
|    ├── 14
|    └── 15
├── *ValidationSet*   # Folder with the images of the validation set
|    ├── 00
|    ├── 01
|    ├── ...
|    ├── 14
|    └── 15
|
| Markdown
├── README.md
|
| Liblinear
├── libsvmread.mexw64   # Read
├── libsvmwrite.mexw64   # Write
├── train.mexw64   # File with train function
├── predict.mexw64   # File with predict function
|
| Matlab Script
├── script_no_live.m   # The script
└── script.mlx   # Live script version
```

Folders with * are not included in the repository.

The pretrained networks

In this section we will show which are the pretrained networks that we used in this project.

A pretrained network is a network that has already learned to extract powerful and informative features from natural images. It can be used as a starting point to learn a new task.

Many of the pretrained networks used in this project have been trained with the same ImageNet database. These pretrained networks can classify images into 100 object categories, such as keyboard, mouse, pencil and many animals.

Alex Net

AlexNet is a convolutional neural network that is 8 layers deep. The network has an image input size of 227x227.

To see the structure of the network in matlab you have to put these lines in the command window:

```
net = alexnet;
```

After

```
net.Layers;
OR
analyzeNetwork(net)
```

Res Net 18

The ResNet-18 is a convolutional neural network that is 18 layer deep. The network has an image input size of 244x244.

To see the structure of the network in matlab you have to put these lines in the command window:

```
net = resnet18;
```

After

```
net.Layers;
OR
analyzeNetwork(net)
```

VGG 16

VGG16 is a convolutional neural network that is 16 layers deep. The network has an image input size of 224x224.

To see the structure of the network in matlab you have to put these lines in the command window:

```
net = vgg16;
```

After

```
net.Layers;
OR
analyzeNetwork(net)
```

Dataset Organization

The first step to do is to organize the images into folders. The zip file is made by a folder that contains all the images and three csv files:

- **training_set**: that contains all the images to use to train the knn
- **validation_set**: that contains all the images to use to verify the accuracy
- **test_set**: that contains all the images to use to run the algorithm

The first two csv files have 6 columns:

- Image name
- x coordinate
- y coordinate
- u coordinate
- z coordinate
- Class

We used just the first two files and we removed all the coordinates columns, because we don't need the position in which the photo was taken, we need just the name of the file and the class.

For both the training and the validation set, using a bash script file, we divided all the images into folders from 00 to 15 based on their class.

So we have two folders:

1. **ValidationSet**: in which we can find all the validation set images
2. **TrainingSet**: in which we can find all the training set images

The images folder won't be in the repository because the size is too high for github.

The **organized dataset** that we used can be downloaded here.

How the project works

In this section we will explain how the project works.

Variables tuning

In the first part of the code it is possible to configure the code variables. This part is useful to enable or disable some parts of the code, or to choose which classification version or pretrained network to use.

Print configuration

These variables enable or disable some part of the code.

1. Print random images of the training set (0 disabled / 1 enabled)

```
print_training_set = 0;
```

2. Print 12 random images of the test set (0 disabled / 1 enabled). For each of them it shows the number of the image, the prediction of the model and the correct class.

```
print_test_set = 0;
```

3. Print the confusion matrix (0 disabled / 1 enabled).

```
print_conf_matr = 0;
```

Classification Version

It is possible to choose between two classifier versions:

1. Matlab: following the matlab tutorials on how to extract features there is the possibility to use the function *fitcecoc* to create a classifier and then generate the predictions.

```
classification_version = "matlab";
```

2. Liblinear: we use the liblinear library to use linears svm to perform the classification. So, after the conversion of the data to the one compatible with liblinear, we train the model passing the labels and the features. After we perform the predictions using the labels and the features of the test set and the model generated before. At the end we compute the accuracy.

```
classification_version = "liblinear";
```

Network selection

It is possible to select one of the pretrained networks between **AlexNet**, **ResNet-18** and **VGG16**.

```
network = "alexnet";
% network = "resnet";
% network = "vgg"
```

Import the dataset and split the training set

In the second part of the code we will import all the images using *imageDataStore*, a function that automatically labels all the images based on the folder names. The images will be stored into an *ImageDataStore* object.

So we take the validation set images from the folder **ValidationSet** and we store them into an *ImageDataStore* object. The same thing for the training set.

Image resize

The networks require different input sizes, in this section the image will be resized to fit the first input layer. To automatically resize the training and test images before they are input to the network, create augmented image datastores, specify the desired image size, and use these datastores as input arguments to activations.

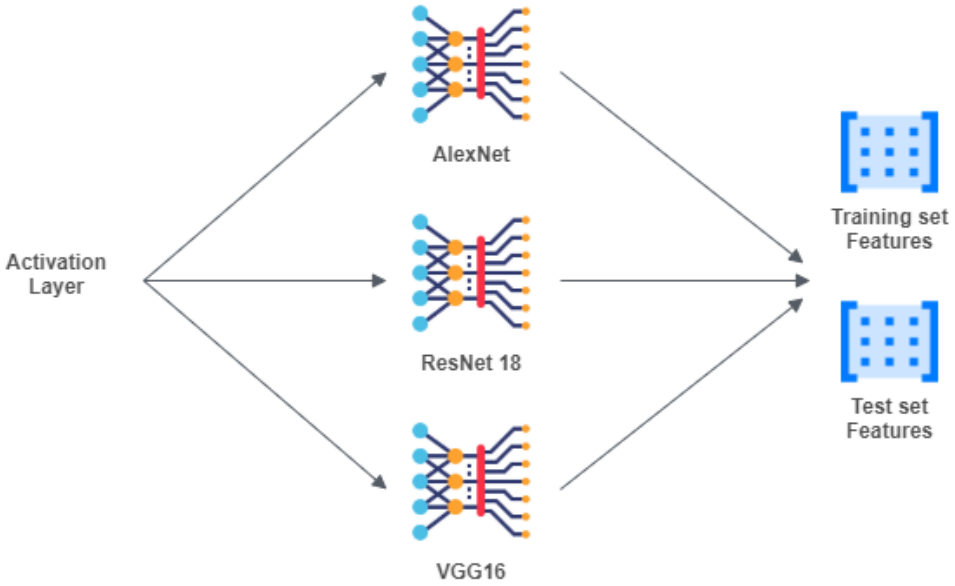
Select the activation layer for the feature extraction and extract the features

The network constructs a hierarchical representation of input images. Deeper layers contain higher-level features, constructed using the lower-level features of earlier layers.

To get the feature representations of the training and test images, we will use activations on different layers depending on the network used.

In our case for **alexnet** is **fc7**, for **resnet18** is **pool5** and for **vgg16** is **fc7**.
This parameter can be changed. Basically we are extracting the feature from the layer before the layer that actually classifies the things.

At the end of this step we will have the features of the training and test sets.



Classification

The next step is to perform the creation of the model using the training set features and labels, and after to perform the classification using the model, the feature of the test set and the labels of the test set. At the end we compute the accuracy.

There are two different versions to do this: **matlab** and **liblinear.**

Matlab Version

Following the matlab tutorials on how to extract features there is the possibility to use the function fitcecoc to create a classifier and then generate the predictions.
At the end it is possible to compute the accuracy.

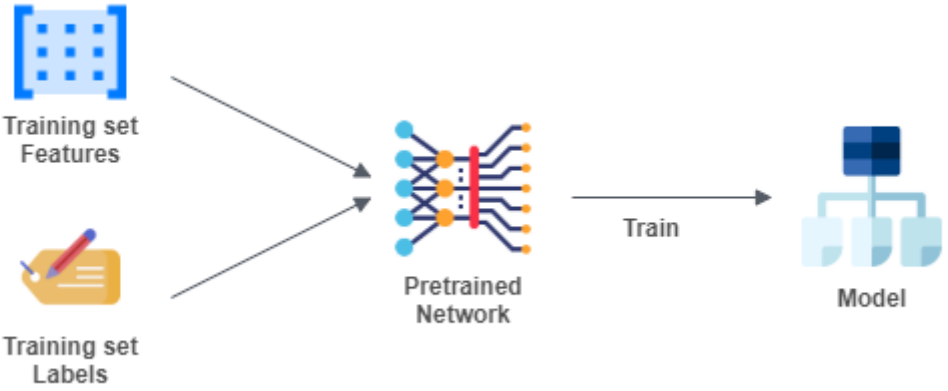
```
classifier = fitcecoc(featuresTrain,YTrain);
YPred = predict(classifier,featuresTest);
accuracy = mean(YPred == YTest);
```

Liblinear Version

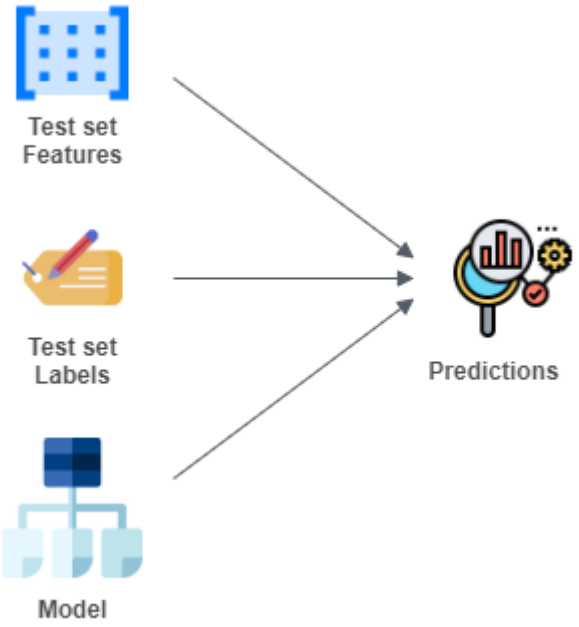
Here we use the liblinear library to use linears svm to perform the classification.

In the first rows of the code there are the conversions of the data to the one compatible with liblinear.

Next we train the model passing the labels and the features. To train the model we insert an option (*-s 2*) to use the **L2-regularized L2-loss support vector classification (primal**). The default is **L2-regularized L2-loss support vector classification (dual)**, but it gives many warnings because it reaches the max number of iterations.



After we perform the prediction using the labels and the features of the test set and the model generated before.



At the end we compute the accuracy.

```
YTrain = double(YTrain(:,1));
YTest = double(YTest(:,1));
featuresTrain = sparse(double(featuresTrain));
featuresTest = sparse(double(featuresTest));
model = train(YTrain, featuresTrain, '-s 2');
YPred = predict(YTest, featuresTest, model);
accuracy = mean(YPred == YTest)
```

How to run the project

In this section we will explain how to run the project

Preliminary steps

1. Clone the repository

```
git clone https://github.com/thisispivi/Deep-Learning-Supermarket
```

2. Open the matlab live script file: ***script.mlx***, or the normal matlab script: ***script_no_live.m***
3. Install the matlab Statistic and Machine Learning Toolbox Add On: Home > Add-On > Statistic and Machine Learning Toolbox
4. Install the matlab Deep Learning Toolbox Model for AlexNet Network Add On: Home > Add-On > Deep Learning Toolbox Model for AlexNet Network
5. Install the matlab Deep Learning Toolbox Model for ResNet-18 Network Add On: Home > Add-On > Deep Learning Toolbox Model for ResNet-18 Network
6. Install the matlab Deep Learning Toolbox Model for VGG-16 Network Add On: Home > Add-On > Deep Learning Toolbox Model for VGG-16 Network
7. Install the Plot Confusion Matrix Add On: Home > Add-On > Plot Confusion Matrix by Vane Tshitoyan

Dataset organization

Here there are two options:

Download our organized dataset

1. Download the organized dataset we used from this link.

Download and manually organize the original dataset

1. Download the original dataset
2. With the help of online tools remove the coordinates columns from the csv files

3. Take the training set images and put them inside a new folder called **TrainingSet** in the root of the project
4. Take the validation set images and put them inside a new folder called **ValidationSet** in the root of the project

Variables configuration

The next step is to configure the variables of the first section. Here there is one of the most important things to do: choose which pretrained network to use to extract the features. To select one you have to uncomment.

```
% network = "alexnet";
% network = "resnet";
% network = "vgg"
```

(Optional)
Configure the other variables

Run the script

The only thing left is to run the script

Test and output analysis

We ran the program with every net and we compared the accuracy, the images correctly classified vs. the number of images and the time elapsed. All the tests have been done using the **liblinear version**.

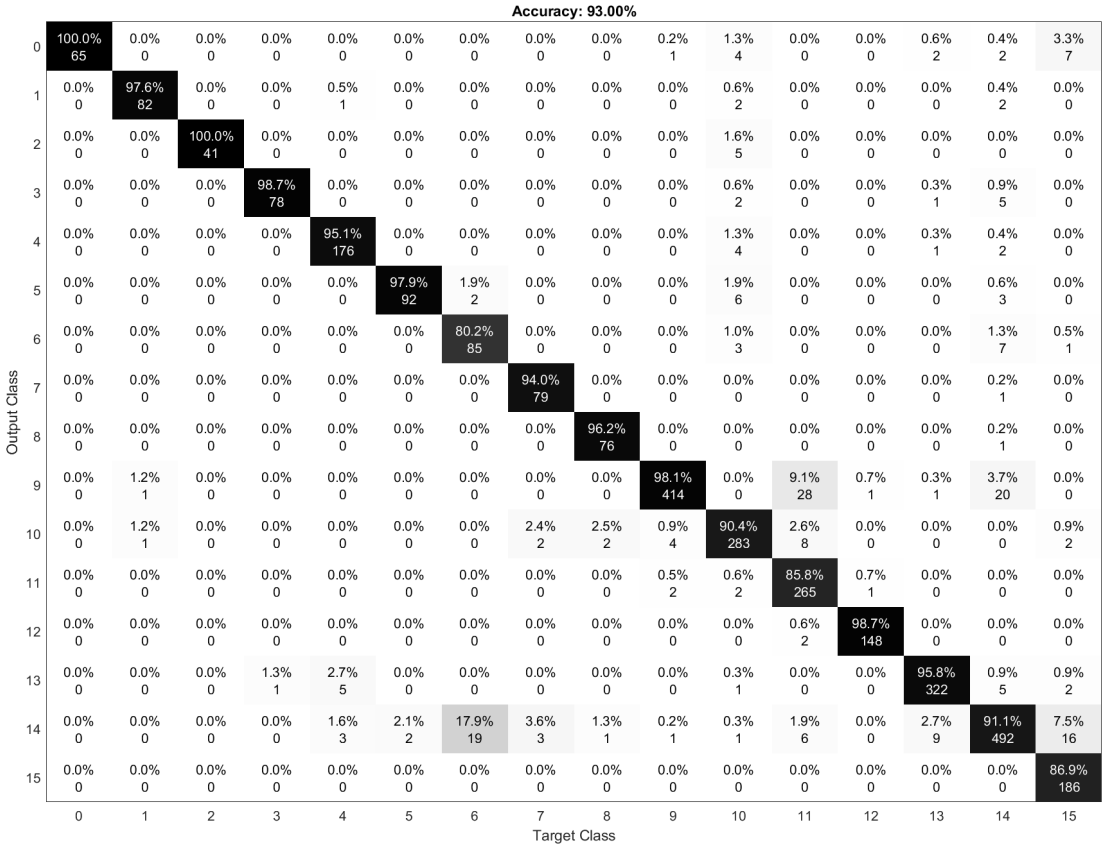
Next for each network we plotted the confusion matrix to understand in which classes there are more errors.

Pretrained Network	Accuracy	Correct classified vs. No. Images	Time elapsed (s)	Time elapsed
AlexNet	93.00%	2884 / 3101	125.51	2 min 5 s
ResNet18	92.52%	2869 / 3101	254.52	4 min 14 s
VGG16	92.33%	2863 / 3101	1595.04	26 min 35 s

As we can see the accuracy is almost the same for each pretrained network, but **AlexNet** is the fastest in terms of time elapsed.

Confusion matrix

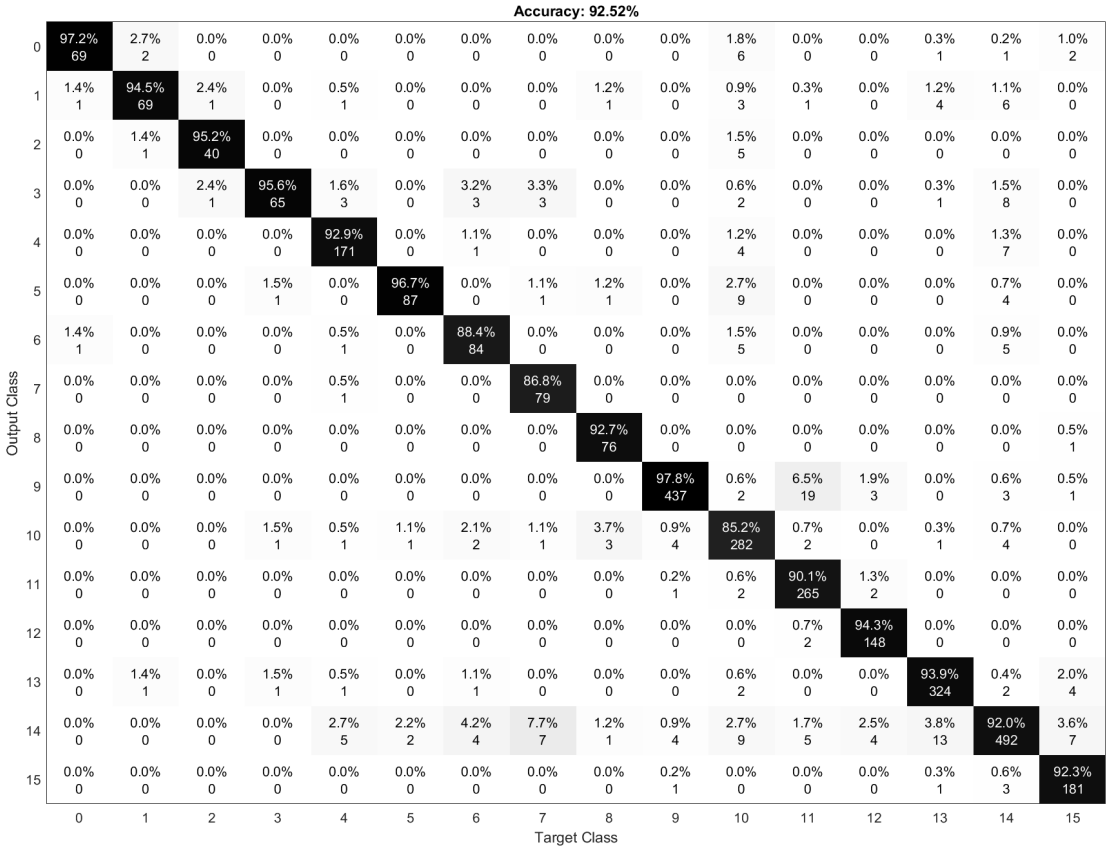
AlexNet



In this confusion matrix we can see that there are many images of the class 6 classified as 14. The main reason is that 14 is a route that has a link with every other class except 10. The errors come from photos that are between 6 and 14, for example a photo of the route 6 in which it is possible to see the route 14 and so on.

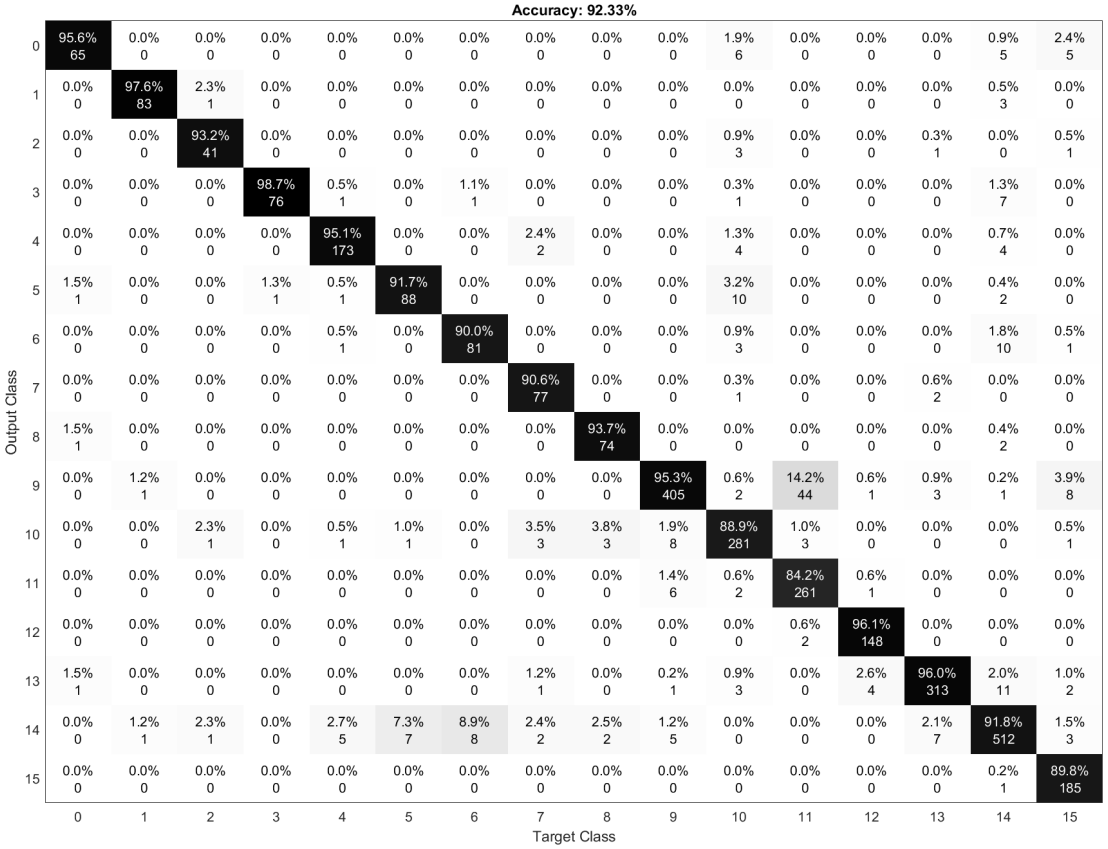
We can also notice images of route 11 classified as 9 and also images of the 15 classified as 14. These errors are caused by the same reason explained before.

ResNet 18



In this confusion matrix we can see images of the class 7 classified as 14 and images of 11 classified as 9. The reason is the same as explained before.

VGG



In this confusion matrix we can see images of the class 5 and 6 classified as 14 and images of 11 classified as 9. The reason is the same as explained before.

Error Analysis

We also analyzed the images misclassified, almost the totality of them comes from images taken between two routes.
For example in some images of route 0 we could see route 15.

Image from the training set of class 00

Image from the validation set of class 15

As we can see the images are almost the same.
These errors could be fixed by removing these images, because they generate many errors.