LUMEN Data Science

BMW

Matej Ciglenečki, Filip Wolf, Borna Katović | Course Title | Date

# The Problem

Computer vision is an area of research that has seen the most growth from the advent of deep learning. Over the past decade, it grew from a niche research area to one of the most widely applicable fields within machine learning. In computer vision, we need to use neural networks to somehow analyze a large number of images, extract some potentially useful information from them, and use that information to classify those images into predefined classes.

The problem we were tasked with solving in this competition falls neatly into this category. Specifically, we were given a list of Google Street View images taken on Croatian roads, 4 for each location and each facing a cardinal direction (north, south, west, east) from the Street View car, and our task is to infer the location where these images were taken, as each image has its coordinates specified. Error is measured using the *haversine* distance measure, which measures the distance between two points over a curved surface, i. e. the surface of the Earth. For every coordinate we infer from the data, the haversine distance to the real coordinate of the image is measured and summed up for all locations. This gives us a total error which will be used to determine how successful our model is compared to others.

On paper, this sounds fairly simple and not unlike many other computer vision tasks. However, this is also a very difficult task, as a country can look very similar over large swathes of land from a car’s perspective and almost feel impossible, even to a human (just think of playing GeoGuesser and telling the difference in which corner of Russia you are). There is a silver lining to this though. Croatia, although small, is very geologically and culturally diverse. Mountains, houses, forests and even fields can look different depending on in which part of the country you are located, giving precedence to the idea that a convolutional neural network could catch these differences. That being said, it is still a difficult problem to solve and requires clever feature engineering and careful network setups in order to work, which we will talk about in the coming chapters.

# Data and Feature Engineering

This problem can be approached from two distinct perspectives. We can either try regressing the image coordinates (a continuous output from the network that will be restricted by the minimum and maximum possible coordinates of the images) or classifying them (assigning areas of the map to different classes). Here, we have chosen the second approach.

## Classification approach

The way we did this is by dividing a map of Croatia into distinct squares representing classes and assigning all images in a square to the corresponding class. We can specify the number of classes before creating the map and thereby create more or less dense lattices that divide the map into classes. By choosing a dense lattice, we can essentially simulate regression, as we have to assign our model output into one out of many possible classes, which is similar to regression, where we have a theoretical infinite number of classes. This has its problems though. There simply aren’t enough images per class for us to effectively train our network with a large number of classes. Another problems arises because of Croatia’s weird shape. Because of it, a lot of the squares mostly end up in other countries territories, with only a side of the square touching the territory of Croatia. Because of this, we end up with squares that don’t contain any classes and we face situations where images are assigned to coordinates that are not within the boundaries of Croatia. This was solved in the following way. In order to solve the first problem, we simply created a “blacklist” of classes that don’t have any images within them and removed them from the possible class assignments our network could perform. We solved the second problem by assigning all images that were classified outside of Croatia to the nearest point that was still within Croatia, but still within the same class. Fortunately, this doesn’t happen too often as the images are very evenly distributed both on the Croatian mainland and its numerous islands.

# Model

There are numerous approaches that are considered state-of-the-art at the moment, all using completely different architectures. Because of readily available high performance models that were pretrained on large datasets being the norm, we also followed this route. Originally, we chose the EfficientNet architecture because of it showing both goof performance and having lower system requirements when compared to other approaches. However, after doing some experimenting, we ended up using a version of ResNet instead, as it simply proved more effective. Of course, we also had to modify these architectures in order to make them work with our dataset. There were two modifications we made.

Firstly, we removed the last layer of the network and replaced it with our own classification layer in the form of a simple linear layer that had the appropriate number of classes for our problem. Secondly, because, as we mentioned before, every location contained 4 images, we modified our network to perform its forward operations for the four images separately and then concatenate the outputs before imputing them into the classification layer. We did this after doing some research on what was the best way to compute the outputs of separate, but statistically linked images.

# Training

Due to using pretrained models, we first performed fine-tuning. ResNet was pretrained on ImageNet, a large image dataset with diverse objects. This gives the early layers of ResNet a collection of learned shapes and lines that are relatively similar to our own domain. In addition, our own dataset is much smaller than the number of ResNet parameters. If we were to train ResNet from scratch, we would quickly overfit. By using a model pretrained on a large generic dataset and then fine-tuning only the last layers, we preserve all the fine detail learned by all the early network layers and only overwrite the last layers where we essentially assemble these details into images. However, after performing fine-tuning for a sufficient time, we periodically unlock all the other network layers so that they can be trained along with the last few layers. By using a sufficiently small learning rate, we induce our dataset information into all the network layers without overwriting them.