# GGMR-CNN

Measurement map classifier by Zsolt Vincze

## Introduction

It is a common practice to build solid waste landfills with an insulation membrane layer. The purpose of this high-density polyester (HDPE) layer is to prevent the leakage of leachate water into the ground, polluting the ground water supply. To determine the integrity of this insulation layer, several geophysical monitoring systems are in use around the world. One of these systems is Geoelectro Geophysical Monitoring System (GGMR). The system utilizing a lot of sensors built under the HDPE layer. The outcome of the measurement is an iso-potential map. Experts of this field can determine if a leakage is present on the map or the layer integrity is acceptable. To make the automation of HDPE layer integrity monitoring easier, I created a convolutional neural network (CNN) based classifier, which can evaluate the measurement maps.

## Building the CNN

I began the network development using Keras with TensorFlow backend. First, I started out from a simple cat-dog classifier network, which can be found in most tutorials of this topic. It contained one convolution and one max polling layer, and a hidden layer with 128 neurons. I began to train the network with the given cat and dog dataset, to see how it works. Soon I found out that one convolutional layer is not enough to reach a good validation accuracy, so I considered other CNN models. I found the VGG 19-layer model, but lack of hardware power I had to shrink it. I created a network containing three convolutional blocks, and a hidden layer. The convolutional blocks build up from three convolutional layers and a max polling layer. The first block generates 32 feature maps and every other block doubles the amount of feature maps. The hidden layer contains 512 neurons. The activation function for these layers is a rectified linear unit (ReLU). The output layer using a sigmoid function because the classifier output is binary.

## Creating the dataset

The available maps were generated in Surfer. First, I had to convert the surfer plots into jpeg format and cut out the relevant parts. Then I rotated the pictures and flipped them vertically and rotated the flipped pictures using a self-made python program. With these transformations I gained eight training pictures from one original. It was necessary because there was a limited amount of maps available and I needed to enlarge the training dataset. I selected one hundred pictures showing leakage and another one hundred for the full integrity. Then I selected twenty new pictures for both categories. In the end I had 1600 training and 336 validation images.

## Training and evaluation

During the training of the network, I met the problem of overfitting. To reduce overfitting, I used image augmentation and Dropout function. For image augmentation I used a function which created 32 of slightly modified image from the original picture. These 32 picture packages became one batch for the training. For dropout, I used small amounts (0.25) in between the convolutional blocks, and a bigger (0.5) after the hidden layer. During the trainings I observed that the validation accuracy tended to oscillate a bit. Because of it, I used ModelCheckpoint callback function to save the model with the best validation accuracy. Keras initialise model weights randomly, so I started the whole training process several times, to see if I can get better accuracy with each try. With these methods I was able to reach 97.9% validation accuracy.