DEPARTMENT OF LANDSCAPE
WATER CONSERVATION

Map digitalisation using multi-spatial resolution approach

ING. ADAM TEJKL

ING. ADAM BABULJAK

CZECH TECHNICAL UNIVERSITY IN PRAGUE

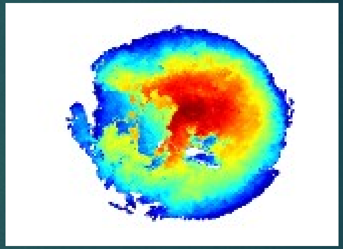
FACULTY OF CIVIL ENGINEERING

THE DEPARTMENT OF LANDSCAPE WATER CONSERVATION

PRAGUE, CZECHIA

ADAM.TEJKL@FSV.CVUT.CZ

Desna 2023



Motivation

- Tedious and slow process of manual identification of different map elements
- Training data will be created as a part of manual map digitalization
- Map fully digitalized and georeferenced
- Software and computing power are easily accessible

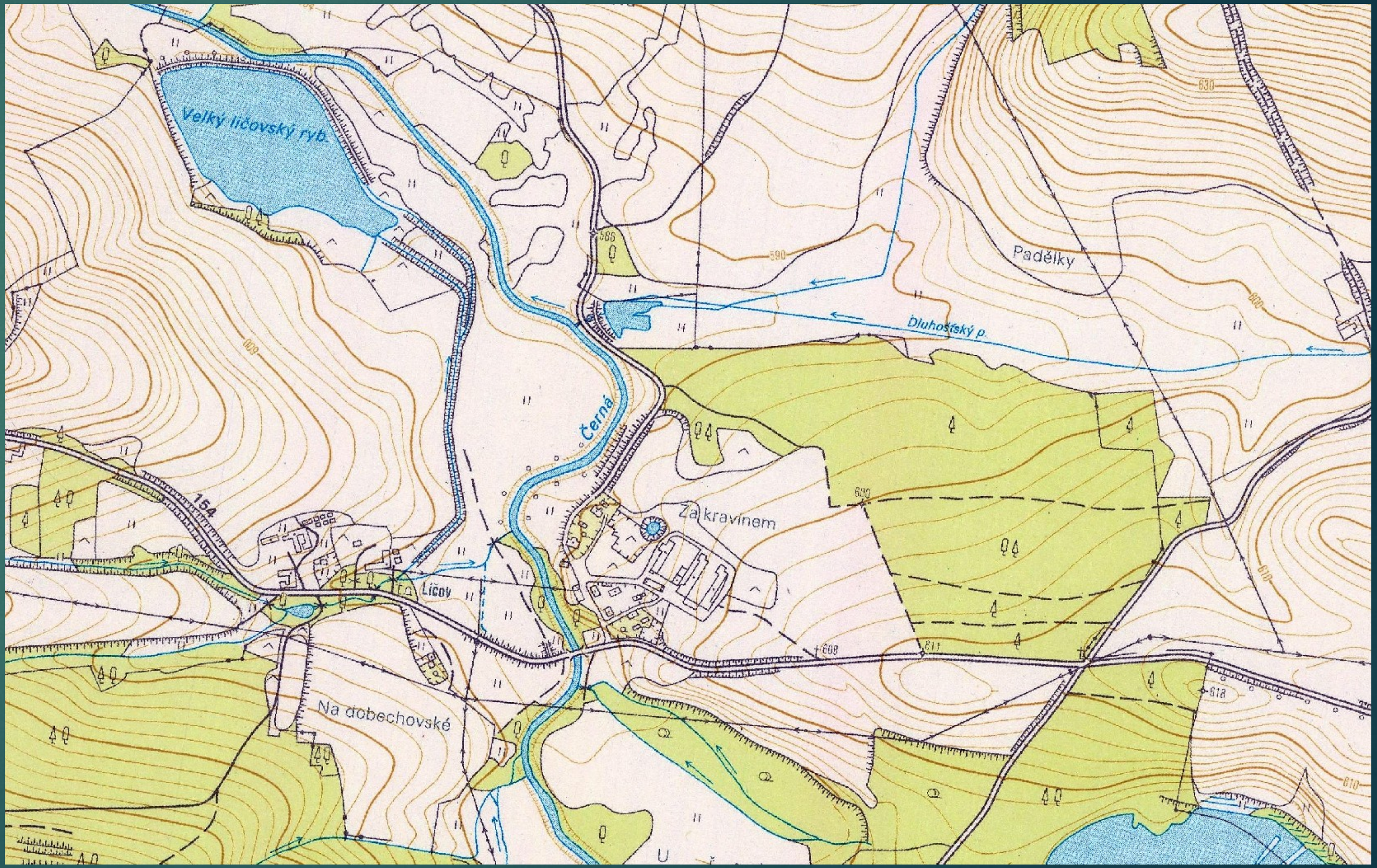
Goals

- Identify several classes of map symbology
  - Small Roads
  - Roads
  - Railway
  - Pasture
  - Dashed Roads
  - Wetland
  - Meadow
  - Rest
  - Boundaries
- Loading of classified raster back to ArcGIS
- Retrainable model

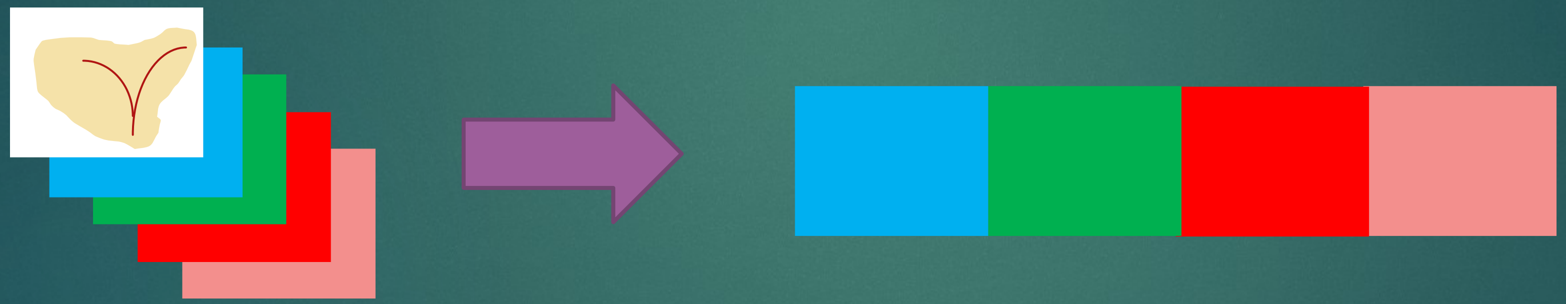
Methodology

- Manual creation of training data
- Creation of multiband composite

- Creation of mosaics from multiband composite segments and their export
- Training of the model
- Use of the model
- Import of the results back to GIS

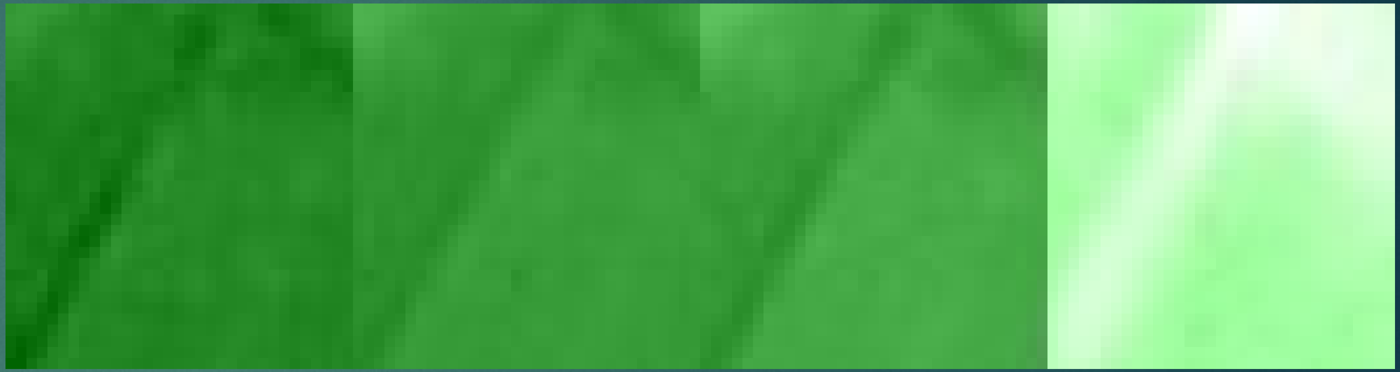


Creation of mosaics from multiband composite segments and their export



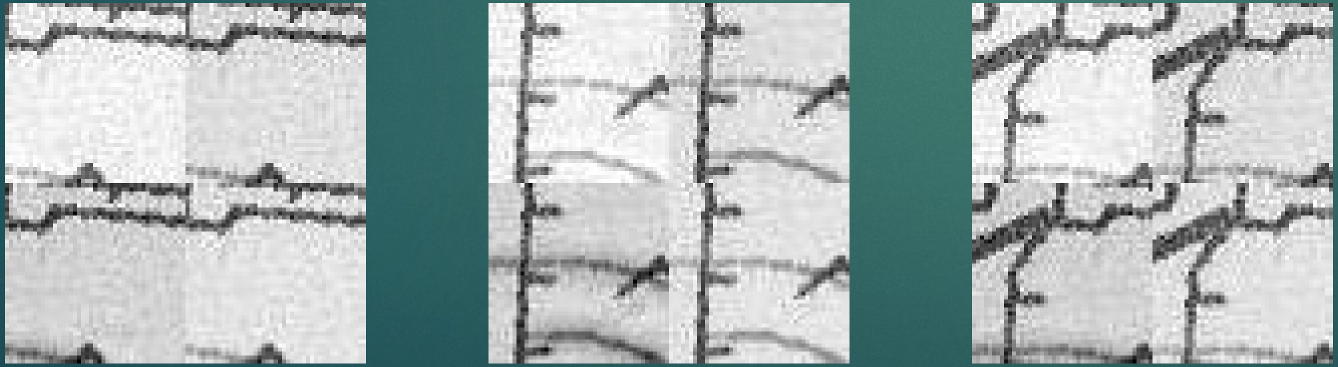
Mosaics from segments

- Variable size
- B, G, R
- Standardize [0, 1]
- Square shift
  - Right
  - Down
  - Down and right
- Saved as jpg format



Segment shift

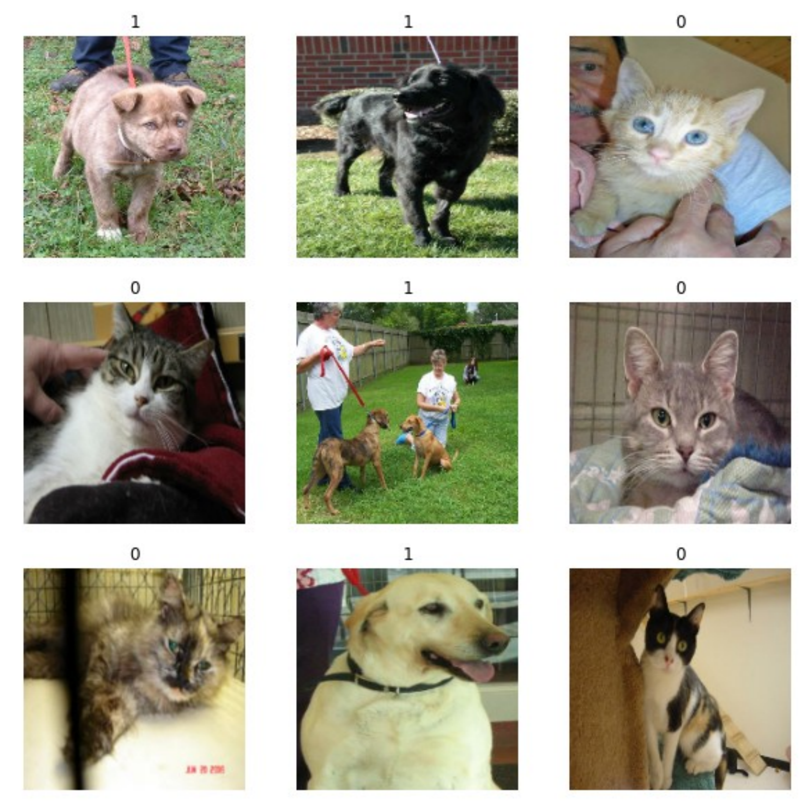
- 4 sets of mosaics are created from one scene
- Doubles the precision of the output
- Classification precision = Pixel size of segment



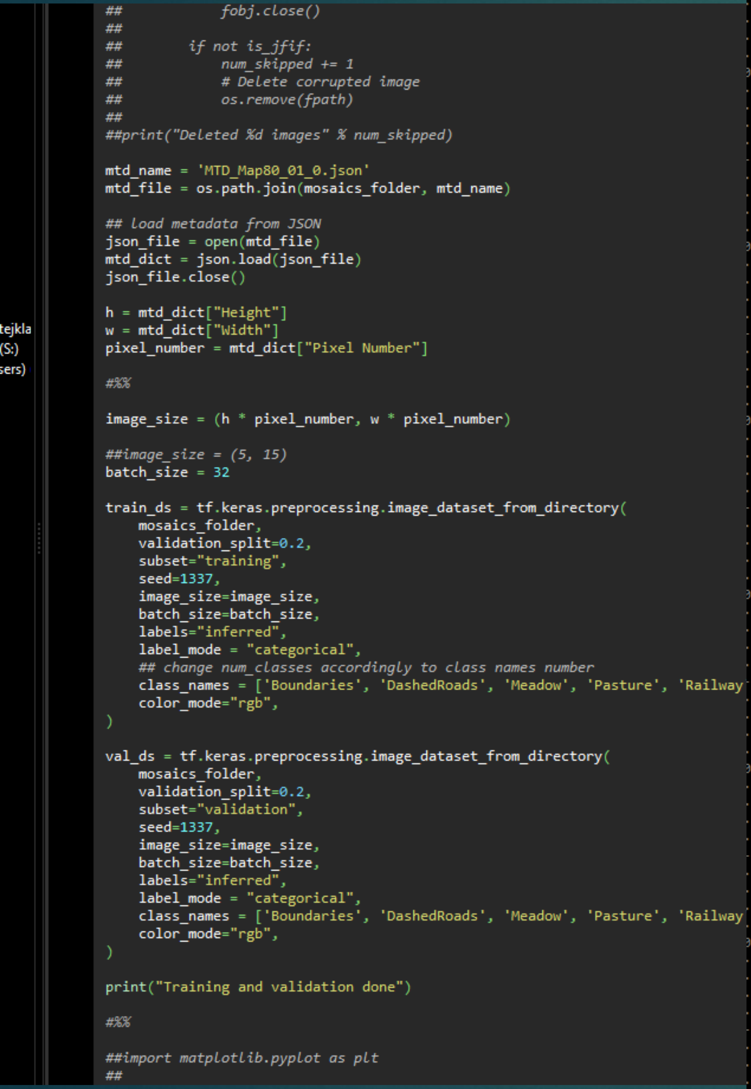
Model

- Python, Tensorflow, Keras
- Deep learning API
- Flexible, Simple workflow, Powerful
- Used by NASA, YouTube, Waymo (Chollet 2021)

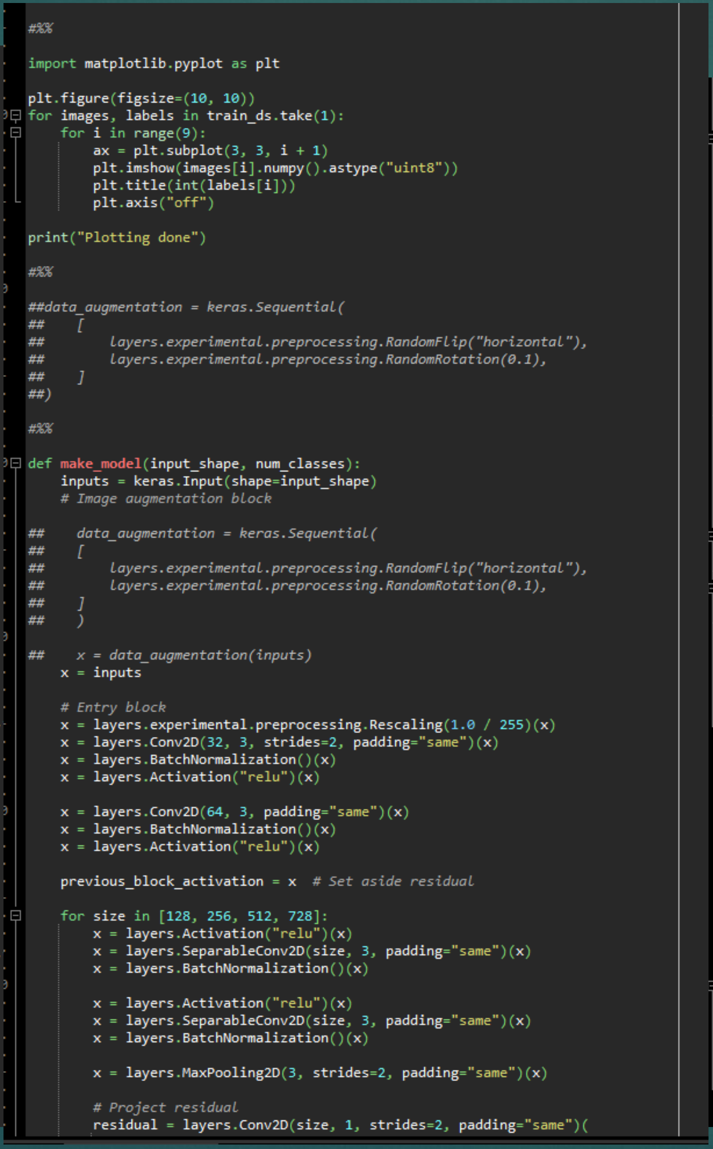
- Kaggle Cats vs Dogs used as a foundation
- 23 410 pictures of cats and dogs
- Binary classification



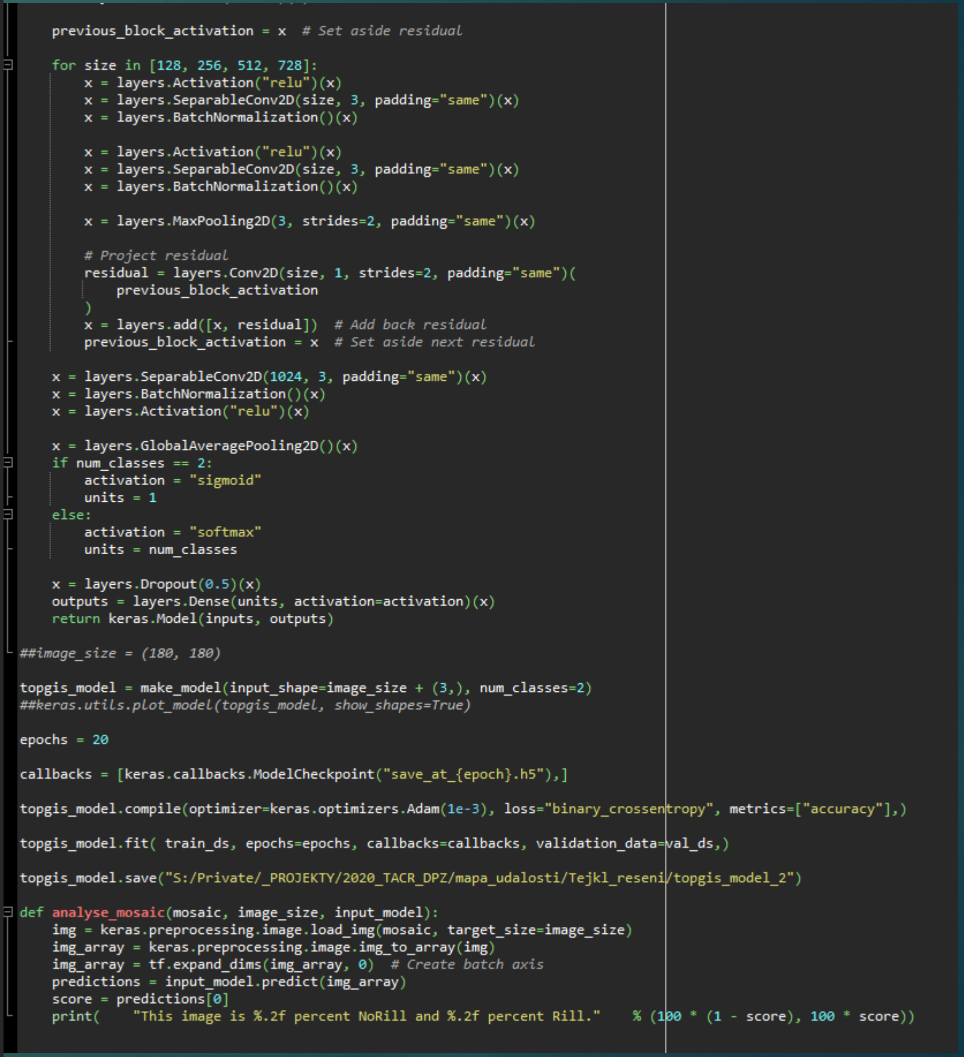
Model



```
##            fobj.close()
##
##        if not is_jfif:
##            num_skipped += 1
##            # Delete corrupted image
##            os.remove(fpath)
##
##print("Deleted %d images" % num_skipped)

mtd_name = 'MTD_Map80_01_0.json'
mtd_file = os.path.join(mosaics_folder, mtd_name)

## load metadata from JSON
json_file = open(mtd_file)
mtd_dict = json.load(json_file)
json_file.close()

h = mtd_dict["Height"]
w = mtd_dict["Width"]
pixel_number = mtd_dict["Pixel Number"]

#%%

image_size = (h * pixel_number, w * pixel_number)

##image_size = (5, 15)
batch_size = 32

train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    mosaics_folder,
    validation_split=0.2,
    subset="training",
    seed=1337,
    image_size=image_size,
    batch_size=batch_size,
    labels="inferred",
    label_mode = "categorical",
    ## change num_classes accordingly to class names number
    class_names = ['Boundaries', 'DashedRoads', 'Meadow', 'Pasture', 'Railway
    color_mode="rgb",
)

val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    mosaics_folder,
    validation_split=0.2,
    subset="validation",
    seed=1337,
    image_size=image_size,
    batch_size=batch_size,
    labels="inferred",
    label_mode = "categorical",
    class_names = ['Boundaries', 'DashedRoads', 'Meadow', 'Pasture', 'Railway
    color_mode="rgb",
)

print("Training and validation done")

#%%

##import matplotlib.pyplot as plt
##
```

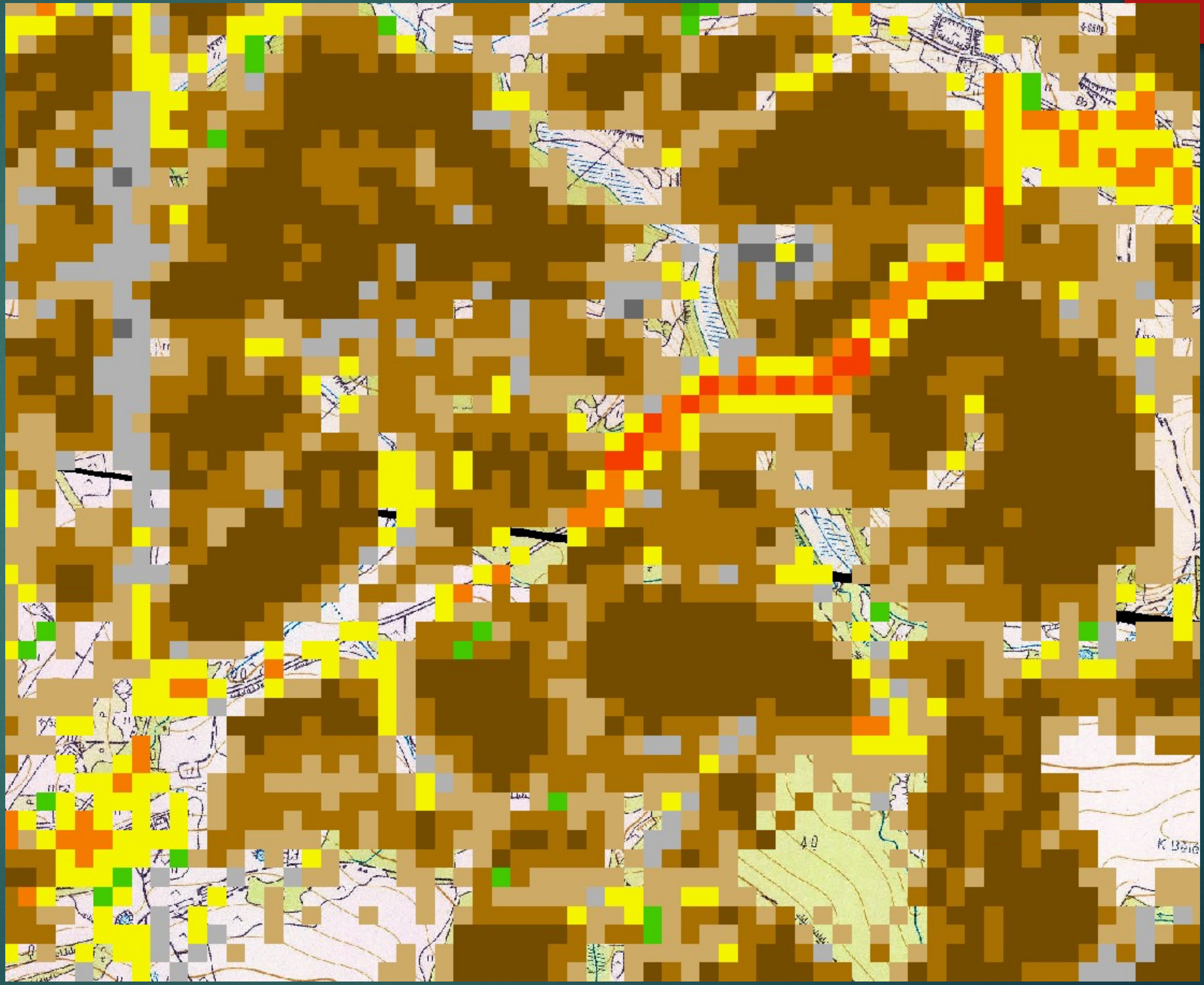


```
#%%

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(int(labels[i]))
        plt.axis("off")

print("Plotting done")

#%%

##data_augmentation = keras.Sequential(
##    [
##        layers.experimental.preprocessing.RandomFlip("horizontal"),
##        layers.experimental.preprocessing.RandomRotation(0.1),
##    ]
##)

#%%

def make_model(input_shape, num_classes):
    inputs = keras.Input(shape=input_shape)
    # Image augmentation block

##    data_augmentation = keras.Sequential(
##    [
##        layers.experimental.preprocessing.RandomFlip("horizontal"),
##        layers.experimental.preprocessing.RandomRotation(0.1),
##    ]
##    )

##    x = data_augmentation(inputs)
    x = inputs

    # Entry block
    x = layers.experimental.preprocessing.Rescaling(1.0 / 255)(x)
    x = layers.Conv2D(32, 3, strides=2, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation("relu")(x)

    x = layers.Conv2D(64, 3, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation("relu")(x)

    previous_block_activation = x  # Set aside residual

    for size in [128, 256, 512, 728]:
        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(size, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(size, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

        x = layers.MaxPooling2D(3, strides=2, padding="same")(x)

        # Project residual
        residual = layers.Conv2D(size, 1, strides=2, padding="same")(
```



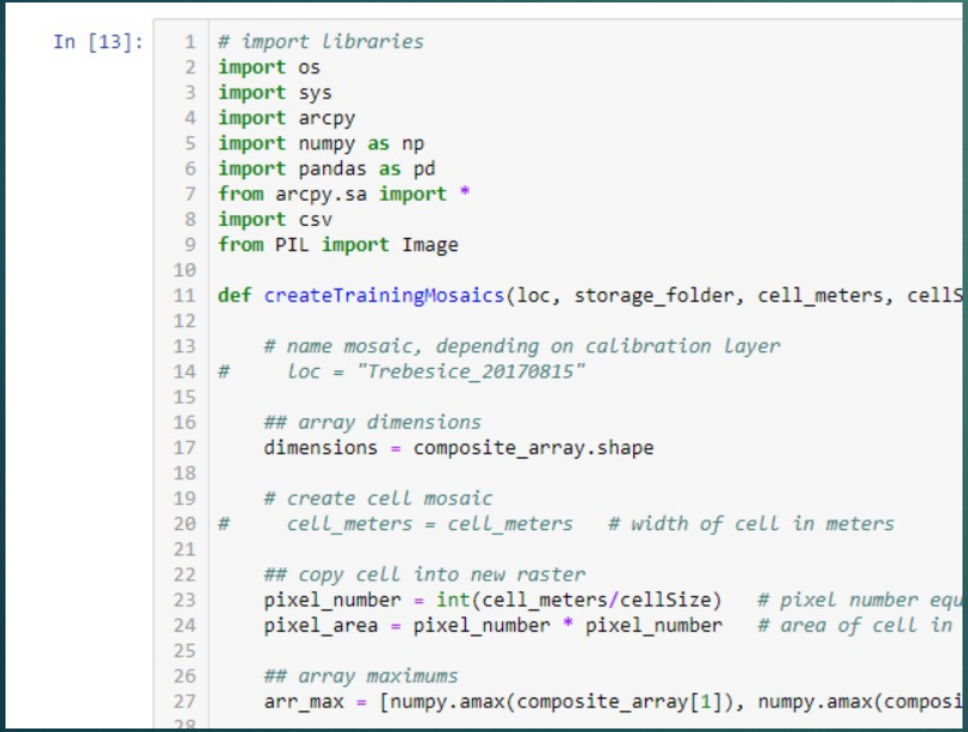
```
    previous_block_activation = x  # Set aside residual

    for size in [128, 256, 512, 728]:
        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(size, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

        x = layers.Activation("relu")(x)
        x = layers.SeparableConv2D(size, 3, padding="same")(x)
        x = layers.BatchNormalization()(x)

        x = layers.MaxPooling2D(3, strides=2, padding="same")(x)

        # Project residual
        residual = layers.Conv2D(size, 1, strides=2, padding="same")(
            previous_block_activation
        )
        x = layers.add([x, residual])  # Add back residual
        previous_block_activation = x  # Set aside next residual

    x = layers.SeparableConv2D(1024, 3, padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.Activation("relu")(x)

    x = layers.GlobalAveragePooling2D()(x)
    if num_classes == 2:
        activation = "sigmoid"
        units = 1
    else:
        activation = "softmax"
        units = num_classes

    x = layers.Dropout(0.5)(x)
    outputs = layers.Dense(units, activation=activation)(x)
    return keras.Model(inputs, outputs)

##image_size = (180, 180)

topgis_model = make_model(input_shape=image_size + (3,), num_classes=2)
##keras.utils.plot_model(topgis_model, show_shapes=True)

epochs = 20

callbacks = [keras.callbacks.ModelCheckpoint("save_at_{epoch}.h5"),]

topgis_model.compile(optimizer=keras.optimizers.Adam(1e-3), loss="binary_crossentropy", metrics=["accuracy"],)

topgis_model.fit( train_ds, epochs=epochs, callbacks=callbacks, validation_data=val_ds,)

topgis_model.save("S:/Private/_PROJEKTY/2020_TACR_DPZ/mapa_udalosti/Tejkl_reseni/topgis_model_2")

def analyse_mosaic(mosaic, image_size, input_model):
    img = keras.preprocessing.image.load_img(mosaic, target_size=image_size)
    img_array = keras.preprocessing.image.img_to_array(img)
    img_array = tf.expand_dims(img_array, 0)  # Create batch axis
    predictions = input_model.predict(img_array)
    score = predictions[0]
    print(    "This image is %.2f percent NoRill and %.2f percent Rill."    % (100 * (1 - score), 100 * score))
```

Results



Evaluation

- Difficult and complex process



```
In [13]:
# import libraries
import os
import sys
import arcpy
import numpy as np
import pandas as pd
from arcpy.sa import *
import csv
from PIL import Image

def createTrainingMosaics(loc, storage_folder, cell_meters, cellS

    # name mosaic, depending on calibration layer
#     loc = "Trebesice_20170815"

    ## array dimensions
    dimensions = composite_array.shape

    # create cell mosaic
#     cell_meters = cell_meters    # width of cell in meters

    ## copy cell into new raster
    pixel_number = int(cell_meters/cellSize)    # pixel number equ
    pixel_area = pixel_number * pixel_number    # area of cell in

    ## array maximums
    arr_max = [numpy.amax(composite_array[1]), numpy.amax(composi
```

ArcGIS Python

Support

- Faculty
- Project No. SS01020366 “Using remote sensing to assess negative impacts of rainstorms“, supported by Technology Agency of the Czech Republic
- Project No. SS01020052 “Utility and risk of irrigation over the Czech Republic in changing climate“, supported by Technology Agency of the Czech Republic

Thanks for your attention

Ing. Adam Tejkl

adam.tejkl@fsv.cvut.cz