Bavarian State Ministry of
Housing, Building and Transport

Esri UC 2019

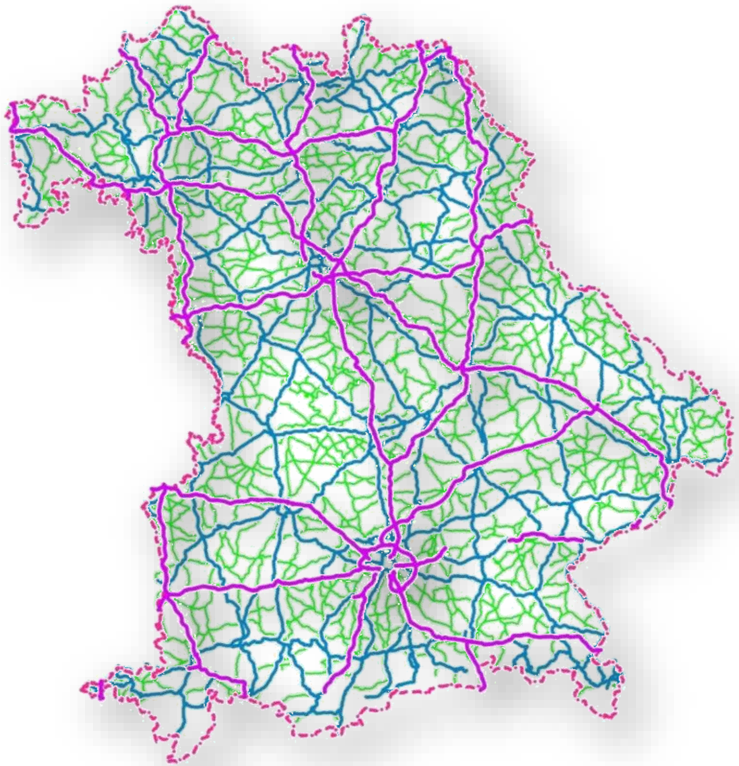
Spatial Intelligence

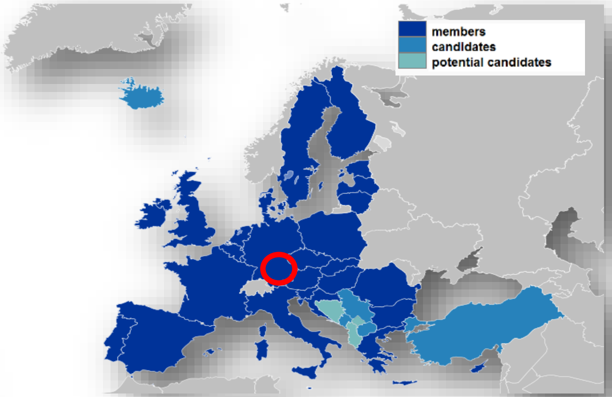
MR Roland Degelmann, CDO
San Diego | 2019-07-10

The Ministry – since march 2018

Staatsminister Dr. Hans Reichhart

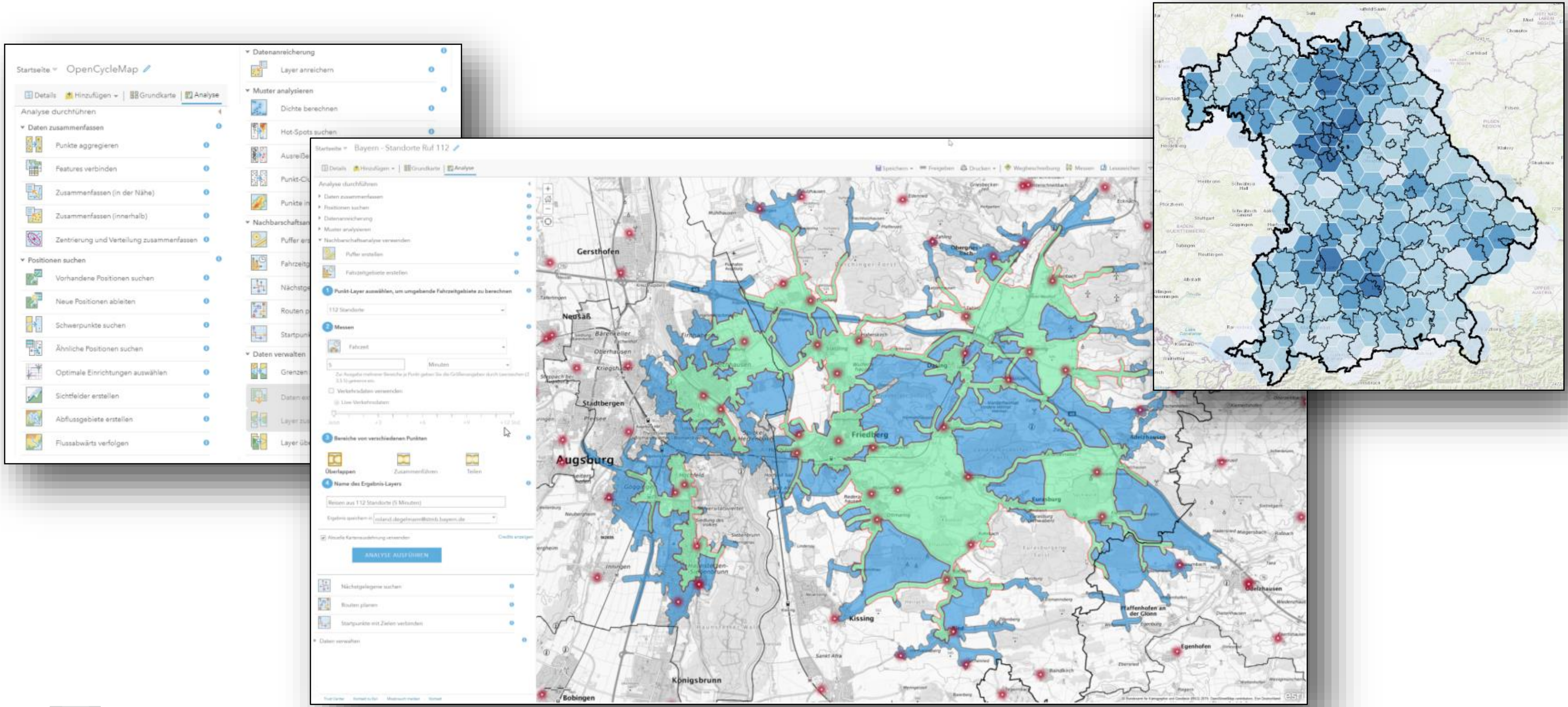
Major Road Network in Bavaria





Motorways:	2.500 km
Federal Roads:	5.900 km
State Roads:	14.100 km
County Roads:	19.000 km

ArcGIS Online – Analysis Tools



AI and Esri ArcGIS

Python

Python is an interpreted, high-level, general-purpose programming language.

Jupyter

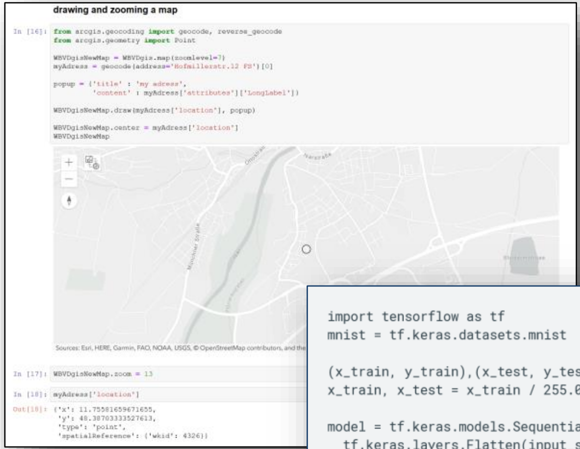
Jupyter Notebook is a web-based interactive computational environment.

ArcGIS API

ArcGIS API for Python is a Python library for working with maps and geospatial data, powered by web GIS.

TensorFlow

TensorFlow is an open source software library for high performance numerical computation.



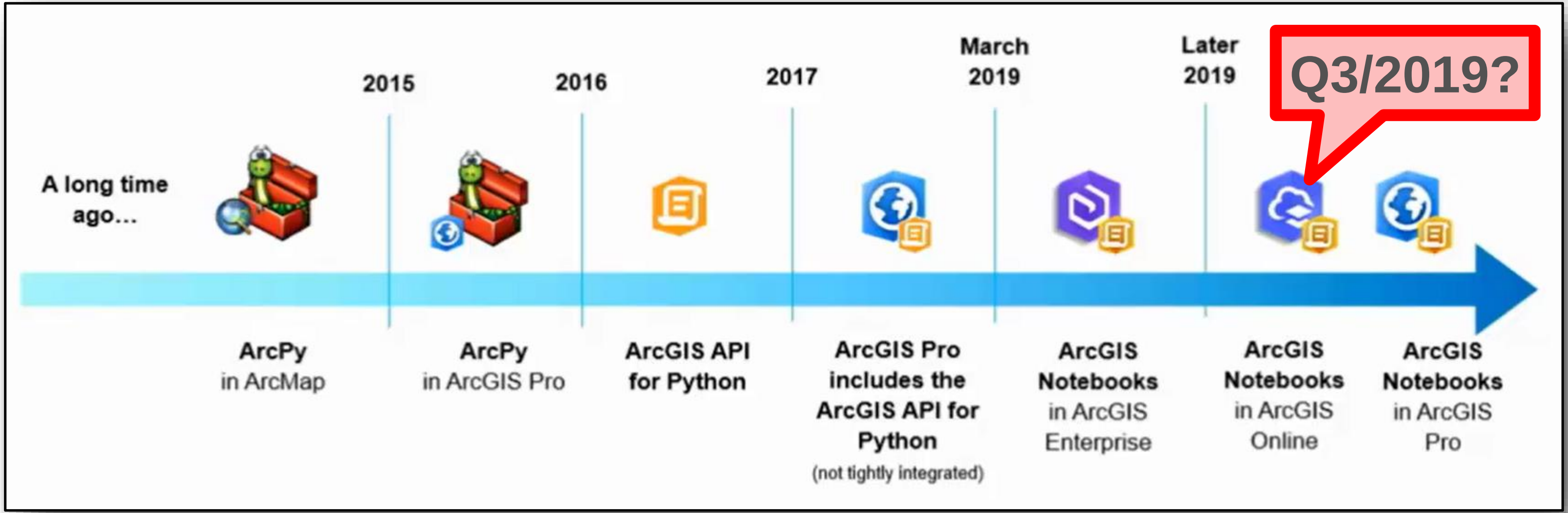

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

www.tensorflow.org

Machine Learning Integration

Source: Esri Inc.

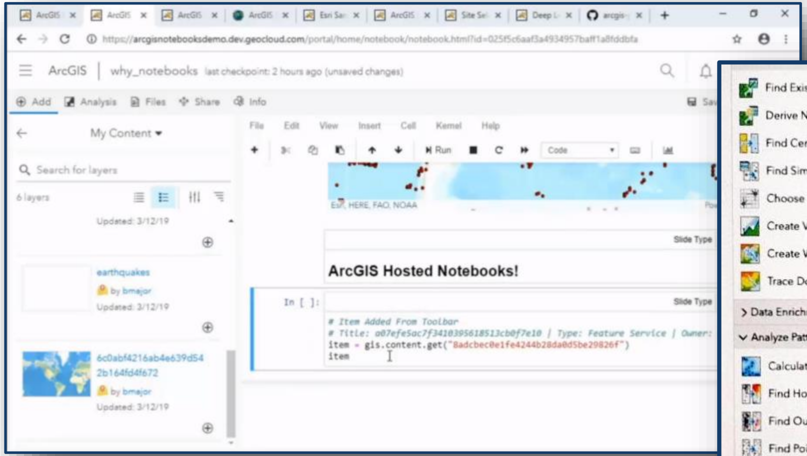
Python across ArcGIS

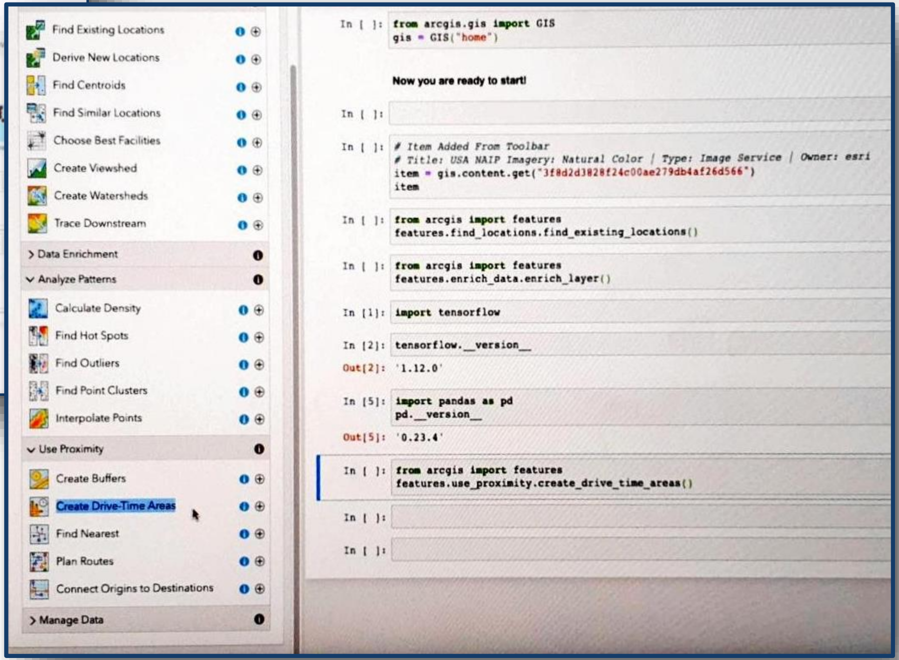


Look for “Esri Get Started with ArcGIS Notebooks”

Source: Esri Inc.

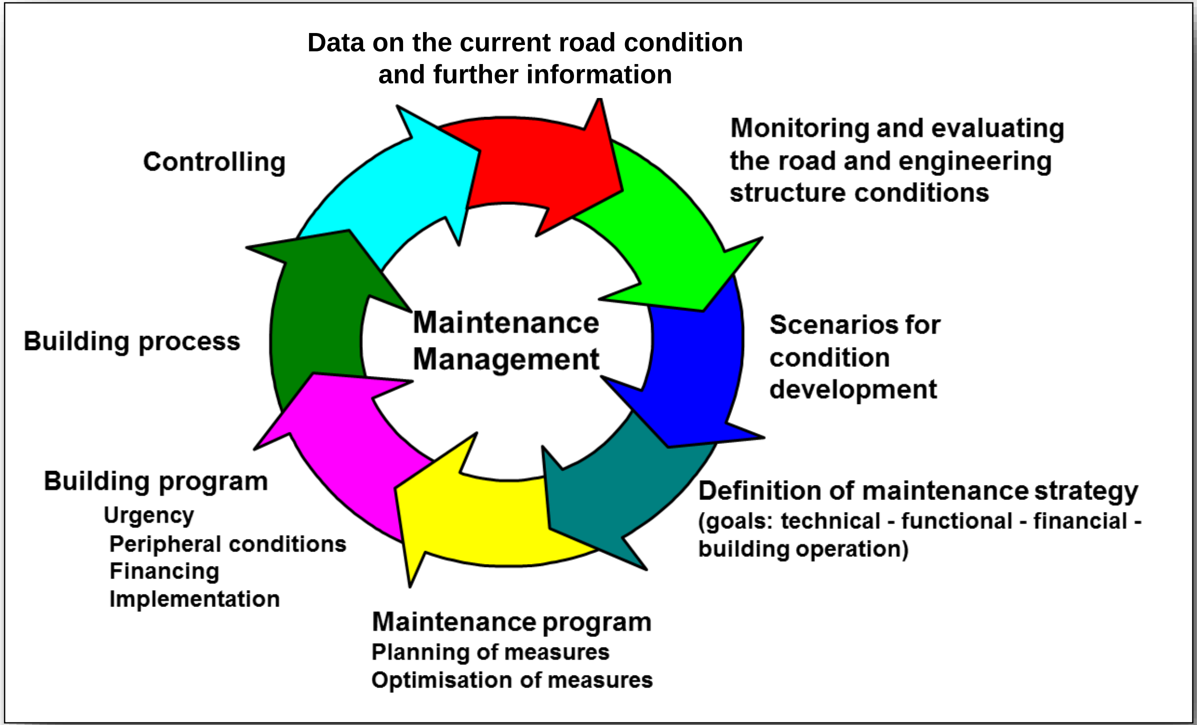
Notebooks in ArcGIS





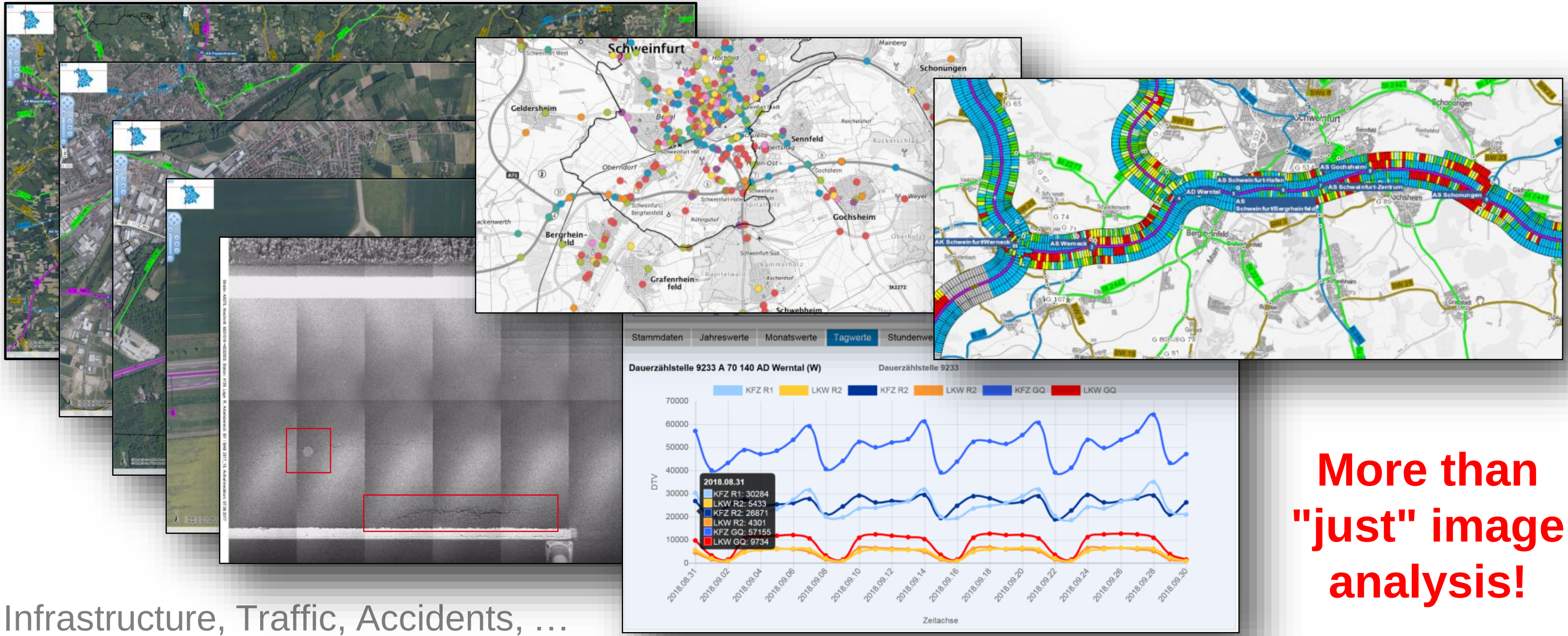
Source: Esri Inc.

Process cycle of Maintenance Management



Predictive Maintenance A70

Database: 2009, 2013, 2017 → Prediction: 2021, 2025

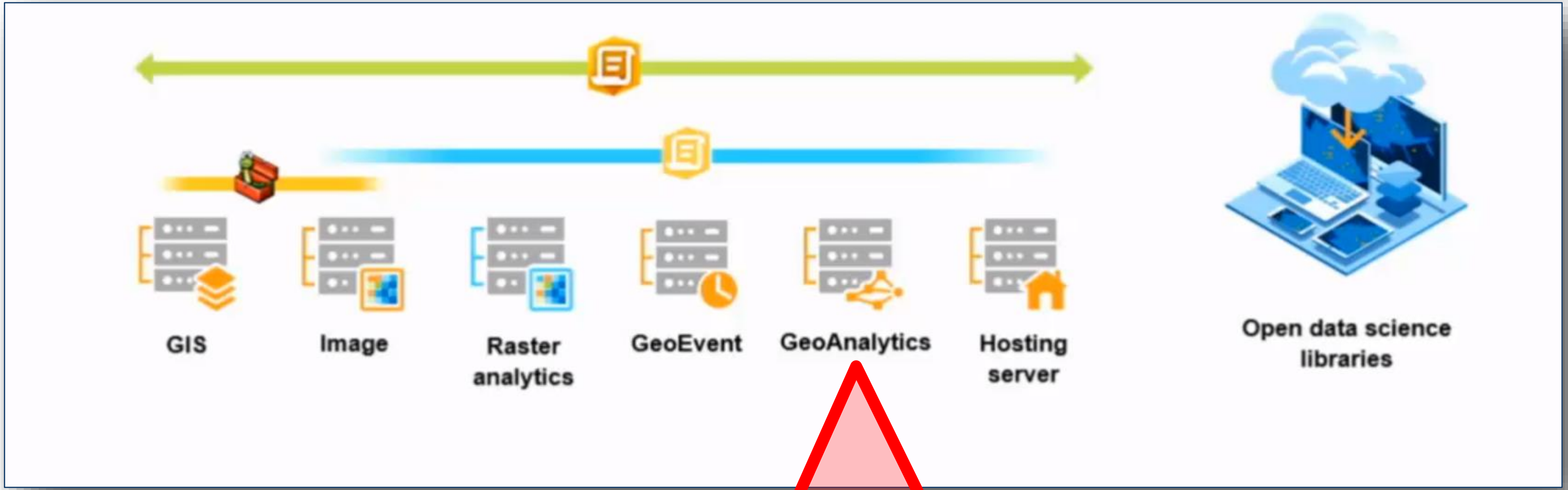


More than "just" image analysis!

Infrastructure, Traffic, Accidents, …

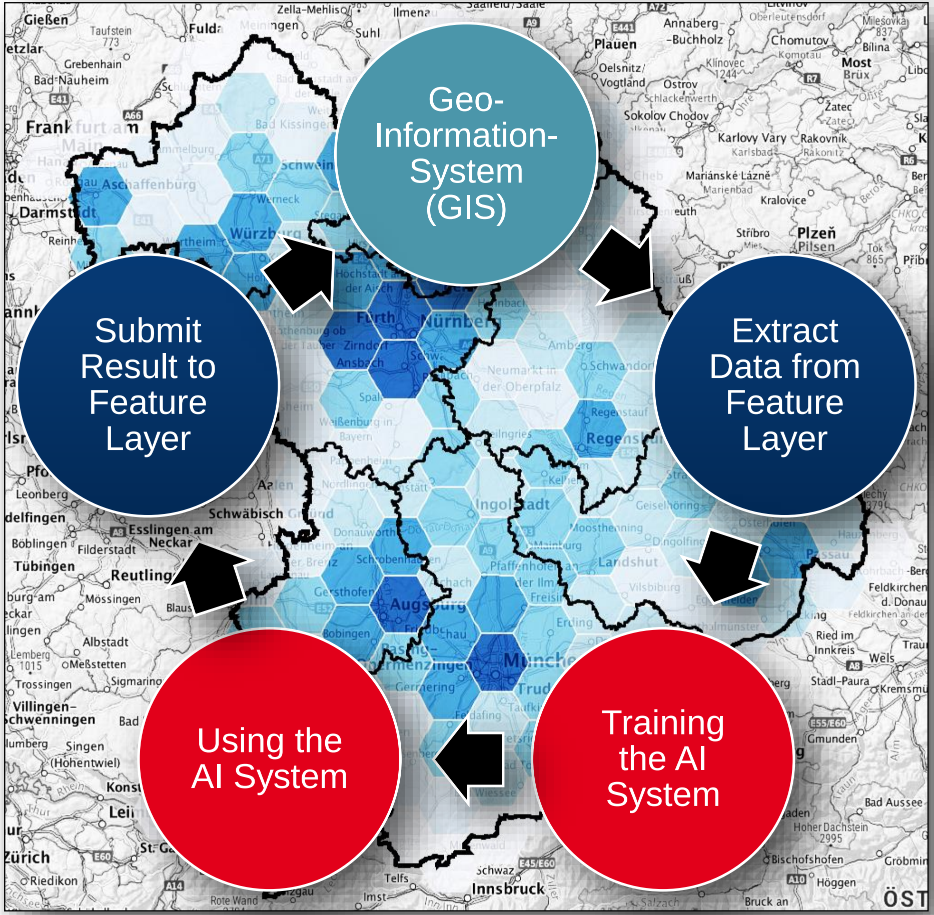
Bavarian State Ministry of
Housing, Building and Transport

Types of analysis



Spatial Intelligence

Source: Esri Inc.



Spatial Intelligence

AI based on Geodata

» Extracting feature-information directly from map-layer

» Deep learning (training / prediction)

» Creating feature-layer directly from prediction

Extracting feature information

```
In [1]: from getpass import getpass
        from arcgis import features
        from arcgis.gis import GIS

        username = 'roland.degelmann@stmb.bayern.de'
        password = getpass()
        WBVDgis = GIS('https://stmb.maps.arcgis.com', username, password)

        ········

In [3]: WBVDgisMapList = WBVDgis.content.search(query = 'BAB', item_type = 'web map')
        WBVDgisMapList

Out[3]: [<Item title:"Erreichbarkeit Anschlussstellen BAB A 8" type:Web Map owner:stefan.schnitzhofer@stmb.bayern.de>,
         <Item title:"BAB Bayern - Erhaltung SI SuS Ergebnisse" type:Web Map owner:roland.degelmann@stmb.bayern.de>,
         <Item title:"BAB Bayern - Fahrbahnzustand" type:Web Map owner:roland.degelmann@stmb.bayern.de>]

In [4]: WBVDgisMapList[2]

Out[4]:
```

BAB Bayern - Fahrbahnzustand
Fahrbahnzustand der Bundesautobahnen in Bayern

Web Map by roland.degelmann@stmb.bayern.de
Last Modified: Mai 12, 2019
0 comments, 2 views

```
In [5]: from arcgis.mapping import WebMap
        from arcgis.features import FeatureLayer

        WBVDgisMapInfo = WBVDgisMapList[2]
        WBVDgisMap = WebMap(WBVDgisMapInfo)

        for layer in WBVDgisMap.layers:
            print(layer.title)

Out[13]: BAYSIS_Anschlussstellen_BAB
         SuS_KT_Geodaten_Ergebnisse
         Verwaltungsgrenzen Bayern - BY_Landkreise_WebMercator
         Verwaltungsgrenzen Bayern - BY_RegBez_WebMercator
         Verwaltungsgrenzen Bayern - BY_Bayern_WebMercator

In [14]: WBVDgisFetaureLayer = FeatureLayer(WBVDgisMap.layers[1].url)
         WBVDgisQueryResult = WBVDgisFetaureLayer.query(where="Bauamt = 'Würzburg'")
         WBVDgisQueryResult.sdf.loc[ :7, ]

Out[14]:
```

	AUN_2009	AUN_2013	AUN_2017	AUN_2021_mean	AUN_2021_sd	AUN_2025_mean	AUN_2025_sd	Abschnitt	Abschnitt_1	BST	...	Strasse_X	Strasse_Y
0	3.46	NaN	3.50	1.872076	0.291191	1.889050	0.166174	100	100	100	...	-135.000167	-74.800692
1	1.57	NaN	1.44	1.829517	0.218778	1.234893	0.167497	100	100	200	...	-135.000167	-74.800692
2	1.30	NaN	1.48	1.916411	0.247385	1.250160	0.158486	100	100	300	...	-135.000167	-74.800692
3	1.77	NaN	2.34	1.820810	0.153464	1.665557	0.151874	100	100	400	...	-135.000167	-74.800692
4	3.80	NaN	1.73	1.415481	0.135631	1.326660	0.154357	100	100	500	...	-135.000167	-74.800692
5	2.00	0.76	1.11	0.726382	0.043498	0.711341	0.100429	120	120	100	...	-135.000167	-74.800692
6	0.71	0.55	1.04	0.825392	0.067845	0.855634	0.137572	120	120	200	...	-135.000167	-74.800692
7	0.84	0.66	3.04	1.552579	0.123425	1.622794	0.166032	120	120	300	...	-135.000167	-74.800692

8 rows × 64 columns

```
In [15]: WBVDgisQueryResult = WBVDgisFetaureLayer.query(where="Bauamt = 'Würzburg'", out_sr='4326')
         print('shape         :', WBVDgisQueryResult.features[0].geometry)
         print('Zustandswert :', WBVDgisQueryResult.features[0].attributes['Zustandswert'])

shape         : {'rings': [[[10.069759374641, 50.0017677879895], [10.0725297326703, 50.0037883409541], [10.07319107674
38, 50.0034125795416], [10.0736583969761, 50.0032012945411], [10.0712314933224, 50.0010048404495], [10.0706036808279,
50.0012936549025], [10.069759374641, 50.0017677879895]]]}
Zustandswert : 3.57
```

Creating feature-layer

In [22]:
```
import pandas as pd
import numpy as np
svzData = pd.read_csv('Data\SVZ-2015-A70_mit-xy.csv', sep = ';', decimal = ',', low_memory = False)
svzData.loc[ :4, ]
```

Out[22]:

	TKZSTNR	Jahr	Strasse	Von	Bis	VNK	NNK	KFZ	SV	LV	Abschnitt	Station	Lon-ZSt	Lat-ZSt
0	59269203	2015	A 70	AK Schweinfurt/Werneck (A 7)	AS Werneck (3)	5926029	5926011	44905	7256	37650	120	2.170	10.108315	50.000537
1	59269004	2015	A 70	AS Werneck (3)	AD Werntal (A 71)	5926011	5926074	48148	7638	40511	140	2.360	10.150259	50.006752
2	59279103	2015	A 70	AD Werntal (A 71)	AS Schweinfurt/Bergrheinfeld (5)	5926074	5927022	45510	6638	38872	160	0.513	10.169445	50.021730
3	59279104	2015	A 70	AS Schweinfurt/Bergrheinfeld (5)	AS Schweinfurt-Hafen (6)	5927022	5927040	42874	6373	36503	180	0.318	10.196808	50.021440
4	59279002	2015	A 70	AS Schweinfurt-Hafen (6)	AS Schweinfurt-Zentrum (7)	5927040	5927023	39874	6037	33836	200	0.706	10.228257	50.022769

In [23]:
```
WBVDgisCsvMap = WBVDgis.map()
WBVDgisCsvMap.basemap = "satellite"
```

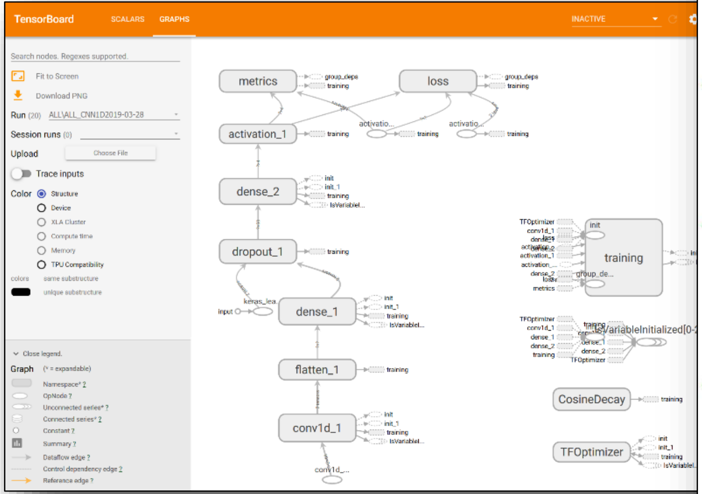
In [24]:
```
thaisymbol = {
 "type": "esriSMS",
 "style": "esriSMSSquare"
}
svzDataArray = np.asarray(svzData)

for i in range(0, len(svzDataArray)) :
    actLocation = {'x' : svzDataArray[i, 12],
                   'y' : svzDataArray[i, 13],
                   'type' : 'point',
                   'spatialReference' : {'wkid': 4326}}
    WBVDgisCsvMap.draw(actLocation, {'title' : 'Abschnitt: ' + str(svzDataArray[i, 3]) +
                                     ' - ' + str(svzDataArray[i, 4]) , 'content' : 'DTV :' +
                                     str(svzDataArray[i, 6]) + ' - SV : ' +
                                     str(svzDataArray[i, 7])}, thaisymbol)
```

In [25]:
```
WBVDgisCsvMap.center = [svzDataArray[7, 13], svzDataArray[7, 12]]
```

In [26]:
```
WBVDgisCsvMap
```

Deep learning

Model structure




```
def LSTM_model(self):
    """ LSTM model with one hidden layer """

    Time_related = 'Yes'
    model_name = 'LSTM'
    model = Sequential()
    model.add(LSTM(self.LSTM_unit(), input_shape=(2, self.input_x.shape[2])))
    self.Activation_variable(model)
    output = self.Compile_and_fit_predict(model,Time_related,model_name)
    K.clear_session()
    print('LSTM Model finished')
    return(output)
```

```
def NN_model (self):
    """ General Neural Network Model with two dense layer
    
    Time_related = 'No'
    model_name = 'NN'
    model = Sequential()
    model.add(Dense(64, activation='relu', input_shape = (
    model.add(Dense(64,activation='relu'))
    self.Activation_variable(model)
    output = self.Compile_and_fit_predict(model,Time_related,
    K.clear_session()
    print('NN Model finished')
    return(output)

def LSTM_model(self):
    """ LSTM model with one hid

    Time_related = 'Yes'
    model_name = 'LSTM'
    model = Sequential()
    model.add(LSTM(self.LSTM_uni
    self.Activation_variable(mo
    output = self.Compile_and_fi
    K.clear_session()
    print('LSTM Model finished'
    return(output)

def CNN1D_model (self):
    """ CNN1D model with one co

    Time_related = 'Yes'
    model_name = 'CNN1D'
    model = Sequential()
    model.add(Conv1D(filters=20
    model.add(Flatten())
    model.add(Dense(self.Dense_
    model.add(Dropout(0.25))
    self.Activation_variable(mo
    output = self.Compile_and_f
    K.clear_session()
    print('CNN1D Model finished
    return(output)

def CNN_LSTM_model (self):
    """ Stacking model of CNN +

    Time_related = 'Yes'
    model_name = 'CNN1D+LSTM'
    model = Sequential()
    model.add(Conv1D(filters=20
    model.add(MaxPooling1D(pool_
    model.add(Dense(self.Dense_
    model.add(Dropout(0.25))
    model.add(LSTM(self.LSTM_un
    self.Activation_variable(mo
    output = self.Compile_and_f
    K.clear_session()
    print('CNN1D+LSTM Model fin
    return(output)
```

```
def Compile_and_fit_predict(self,model,Time_related,model_name):
    """ Model compile and fit

    After the model is build, this function compile and fit the model.
    It includes
        1. learning rate optimizer
        2. data import format
        3. Model prediction
        4. output format

    """
    lr_decayed = tf.train.cosine_decay(0.0001, self.global_step, self.decay_steps)
    opt = tf.train.AdamOptimizer(lr_decayed, epsilon=self.adam_epsilon )

    model.compile(optimizer=opt,loss=self.loss_function_selection(),metrics=[self.coeff_determination] )

    if Time_related == 'Yes':
        X_input = self.input_x
        Y_input = self.train_y
        X_Predict = self.predict_x
    else:
        X_input = self.input_x_NN
        Y_input = self.train_y
        X_Predict = self.predict_x_NN


    model.fit(X_input, Y_input, epochs=10000, batch_size=1024,validation_split=self.split_rate,
                            verbose=0,callbacks=self.callback_function(model_name),shuffle =True)
    yhat = model.predict(X_Predict, verbose=0)

    if self.output_variable == 1:
        dime = 0
    else:
        dime = range(0,self.output_variable)
    return(yhat[:,dime])
```

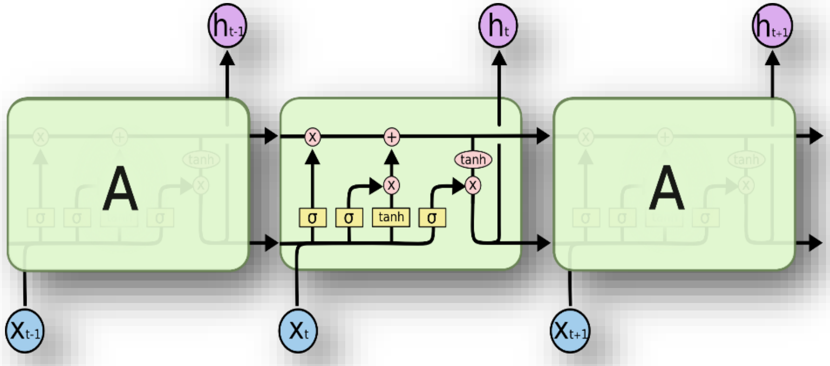
technical implementation: **supper&supper** , Berlin

Usage of neural networks

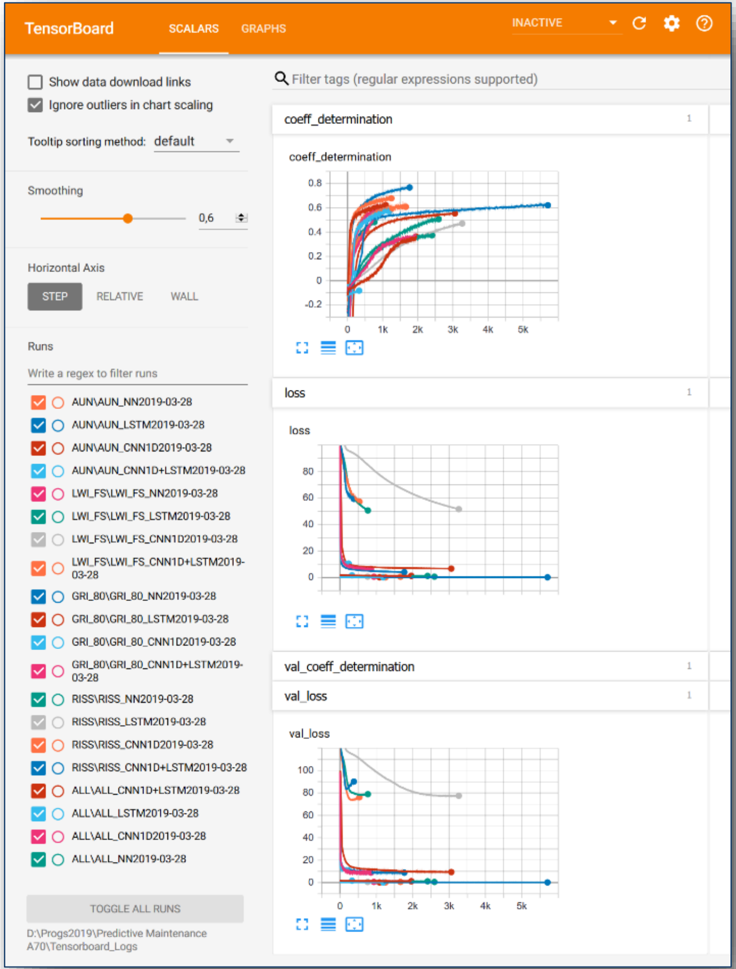
In the current project at StMB multi-variant and spatio-temporal neural networks are applied. The predictions are influenced by the time axis and the geographical position, as well as by the other environment variables.

Used neural networks

- **General Neural Network**
- **Long Short Term Memory (LSTM)**
  - Sequence-to-Sequence Prediction
  - Forgot and update layer from previous point in time
  - 1D- Convolutional Neural Network
  - Discrete time series forecast
- **Stacked CNN-LSTM-Modell**



source: colah.github.io/posts/2015-08-Understanding-LSTMs/

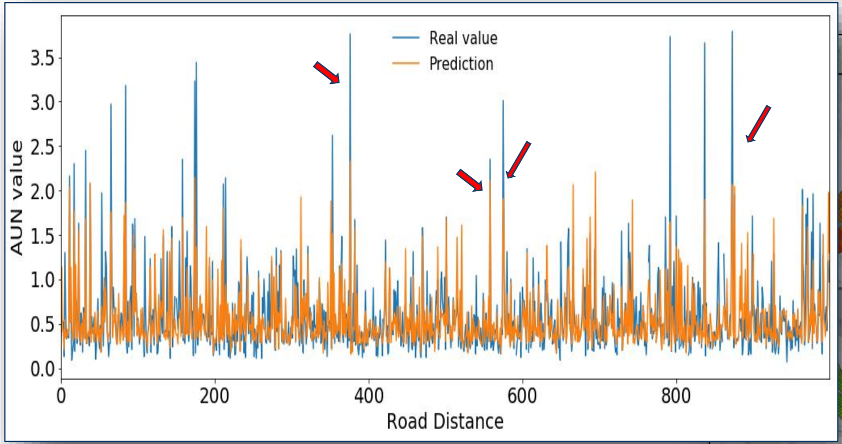


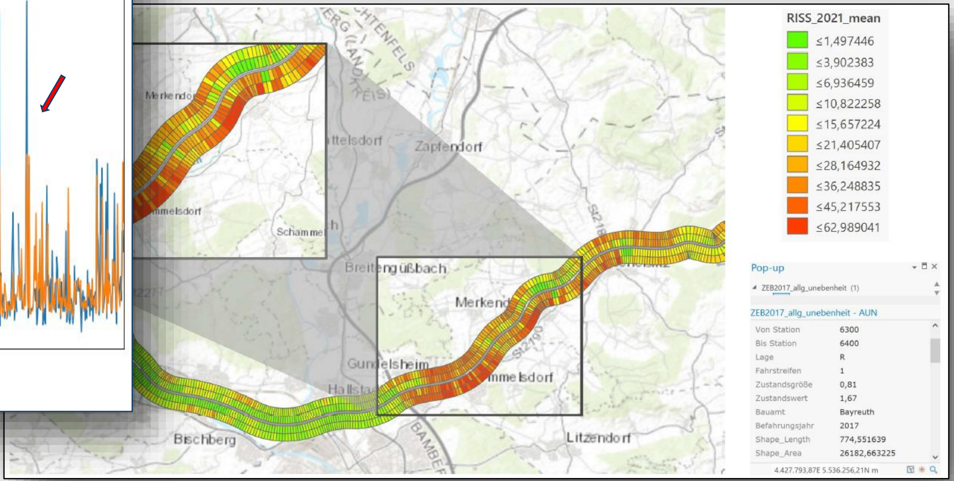
Training

Quality results shown with Tensorboard

Prediction result

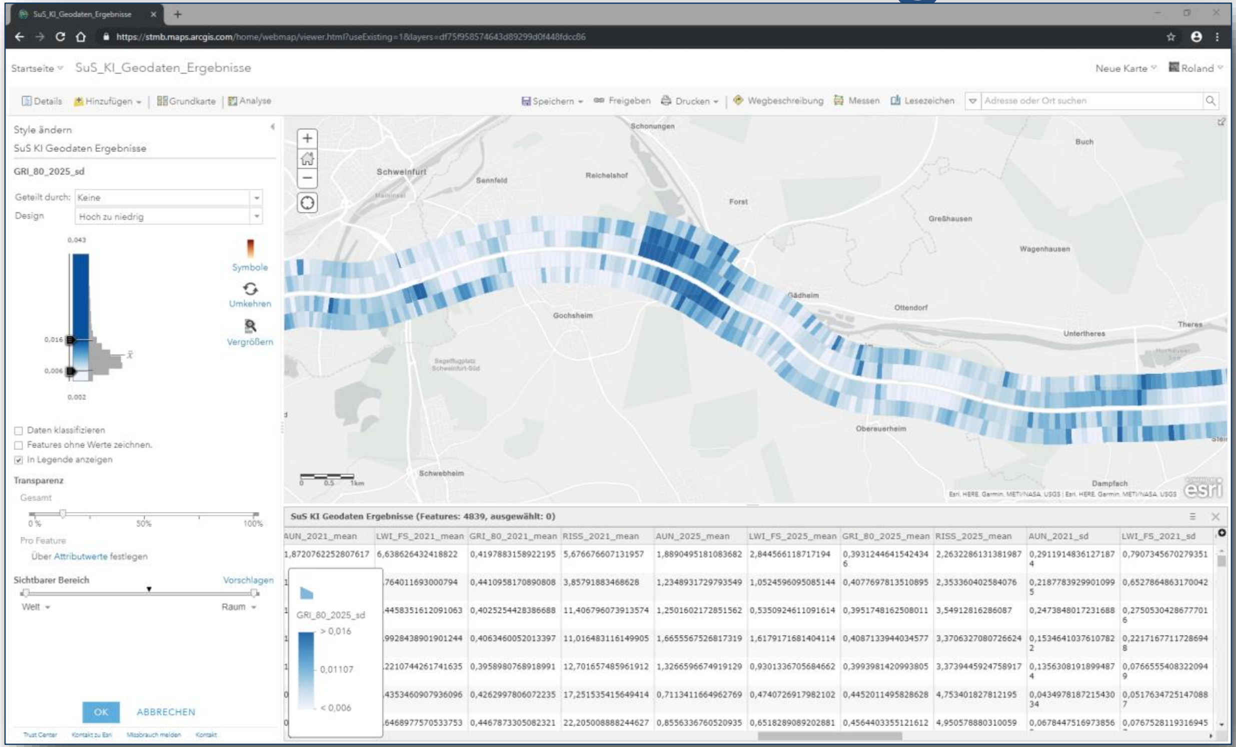
The CNN+LSTM model generally provides good predictions. The variability of the prediction was equal to that of the actual AUN value. Extreme values in the real data are also represented by the model (see red arrows in the left diagram).



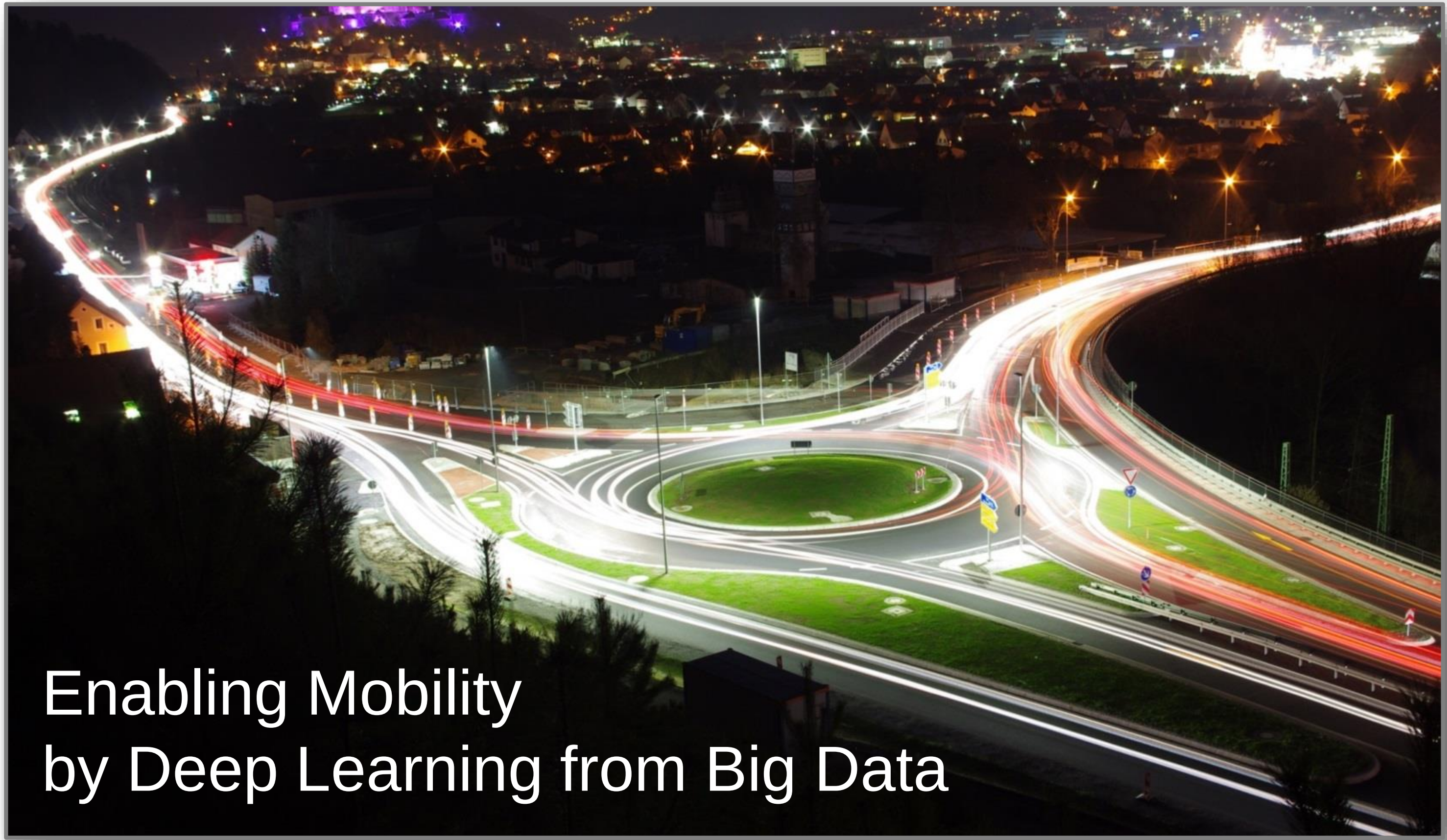


technical implementation: supper & supper, Berlin

ArcGIS Online - Writing results



ArcGIS Online – Using results


Enabling Mobility
by Deep Learning from Big Data