General Remarks:

- Feel free to use the libraries *Matplotlib, Tensorflow, Keras, Pandas, Numpy* & *Scikit-learn.*
- For most questions regarding the implementation, you can find answers in the respective documentation or on *StackOverflow.*
- Make sure to discuss your plots/results and provide reasoning for the choices of your model configurations.

On *TUWEL* you can find the *facemask* dataset, which consists of around 3800 images of people with & without facemasks. For the project, you will implement a Face Mask detection model — which should correctly classify unseen images into *with_mask* and *without_mask.*

Overall the project consists of the **classification**, a **transfer learning** and an **adversarial machine learning** part. For that, prepare the data by rescaling it to a shape of (96, 96, 3) and split it into a training and test dataset according to a 70 to 30 ratio.

1. Face Mask Detection

Your first task is to implement a machine learning model, which detects whether a person is wearing a face mask or not.

$\boxed{1.1}$ Train a Ridge Classifier, a Logistic Regressor, and a SVC from *scikit-learn* on the dataset and evaluate the performance. Briefly evaluate the influence of the available regularization hyperparameters and the effects of different kernels or preprocessing schemes such as standardization or scaling.

$\boxed{1.2}$ Further, obtain the importance of each feature according to the Ridge Classifier and highlight the results graphically for each of the three input channels.

$\boxed{1.3}$ Use *Keras* to implement a neural network which achieves an accuracy on the testset of above 90%. Plot the learning curves and discuss the configuration of your model. Note that you are free in the design of your network.

2. Transfer Learning

In transfer learning, we utilize a pretrained model from a similar domain to enhance the performance of a given regression or classification scheme.

$\boxed{2.1}$ Set up a *MobileNetV2*[1] model in *keras* using the pretrained weights from the imagenet dataset. Subsequently, evaluate the predictions of the network on the *facemask* dataset and analyze the distribution of the assigned labels.

$\boxed{2.2}$ Now, drop the final output layers of the pretrained *MobileNetV2* and process the *facemask* data through the remaining layers. Retrain the *scikit-learn* models from Part 1 on the processed inputs and compare the performance.

[1]See https://keras.io/api/applications/

$\boxed{2.3}$ Implement a neural network consisting of the pretrained *MobileNetV2* model and a newly added output layer suited for the binary *facemask* classification task. Train the last layer of this model while keeping the *MobileNetV2* weights fixed. Compare the performance of the obtained combined model to the neural network from Part 1.

3. Adversarial Machine Learning

In the following, we will examine three variants of **Evasion** attacks on your trained neural network classifier from Part 1. As such, we aim to obtain input samples, for which the trained model will output a desired label. Note that you are **not allowed to alter the weights** of the pretrained model. Make sure to include the generated adversarial inputs in your report.

$\boxed{3.1}$ Start by implementing a neural network, that uses arbitrary Gaussian noise as the input and generates images, that get classified as *with_mask* by your trained network. Note, that this requires that you combine your adversarial generator model with the trained classifier from Part 1.

$\boxed{3.2}$ Further, implement a neural network which takes valid images as inputs and alters them by adding adversarial noise ontop. As such, your network should be trained to flip the labels of the samples from the *facemask* dataset, while at the same time keeping the deviations of the input images minimal. For that, control the $l1$ norm of the additive noise by either including an additional loss or utilizing activity regularization.

$\boxed{3.3}$ Finally assume, that you have complete access to the network under attack and implement the gradient based approach decribed in `https://arxiv.org/abs/1412.6572`. Note, that you can obtain the signs of the gradients with respect to the input samples via *tensorflow*. Repeat the approach for the advanced model based on *MobileNetV2* from Task 2.3.