Monographic Lecture in Mathematics
Solution to Lab 2
Michał Korycki, 229055
Github link: *https://github.com/MatornenSinera/BDA-2ndSemester/tree/master/3rd-semester/Monographic%20Lecture%20In%20Mathematics*

Application of continuous wavelet transform to feature extraction.

Task 1. Prepare four arrays: two for the train/test signals and two for the train/test labels. Each signal has 128 samples and 9 components.

```
LABEL_NAMES = ["Walking", "Walking upstairs", "Walking downstairs", "Sitting", "Standing", "Laying"]

def load_y_data(y_path):
    y = np.loadtxt(y_path, dtype=np.int32).reshape(-1,1)
    # change labels range from 1-6 t 0-5, this enables a sparse_categorical_crossentropy loss function
    return y - 1

def load_X_data(X_path):
    X_signal_paths = [X_path + file for file in os.listdir(X_path)]
    X_signals = [np.loadtxt(path, dtype=np.float32) for path in X_signal_paths]
    return np.transpose(np.array(X_signals), (1, 2, 0))

x_train = load_X_data(direc+file_x_train)
x_test = load_X_data(file_x_test)
y_train = load_y_data("/content/drive/My Drive/Colab Notebooks/Programming/BDA-2ndSemester/3rd-semester/M
y_test = load_y_data("/content/drive/My Drive/Colab Notebooks/Programming/BDA-2ndSemester/3rd-semester/Mo


print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(7352, 128, 9)
(7352, 1)
(2947, 128, 9)
(2947, 1)
```

Comment: As was predicted by the task description, after applying transposition to all files contained within the X – data folder, we got appropriate number of signals consisting of 128 samples and 9 components.

Task 2. Apply CWT to each signal. Use the morlet wavelet (real) and scales from 1 to 128. Each procedure should result in an 128 by 128 image.

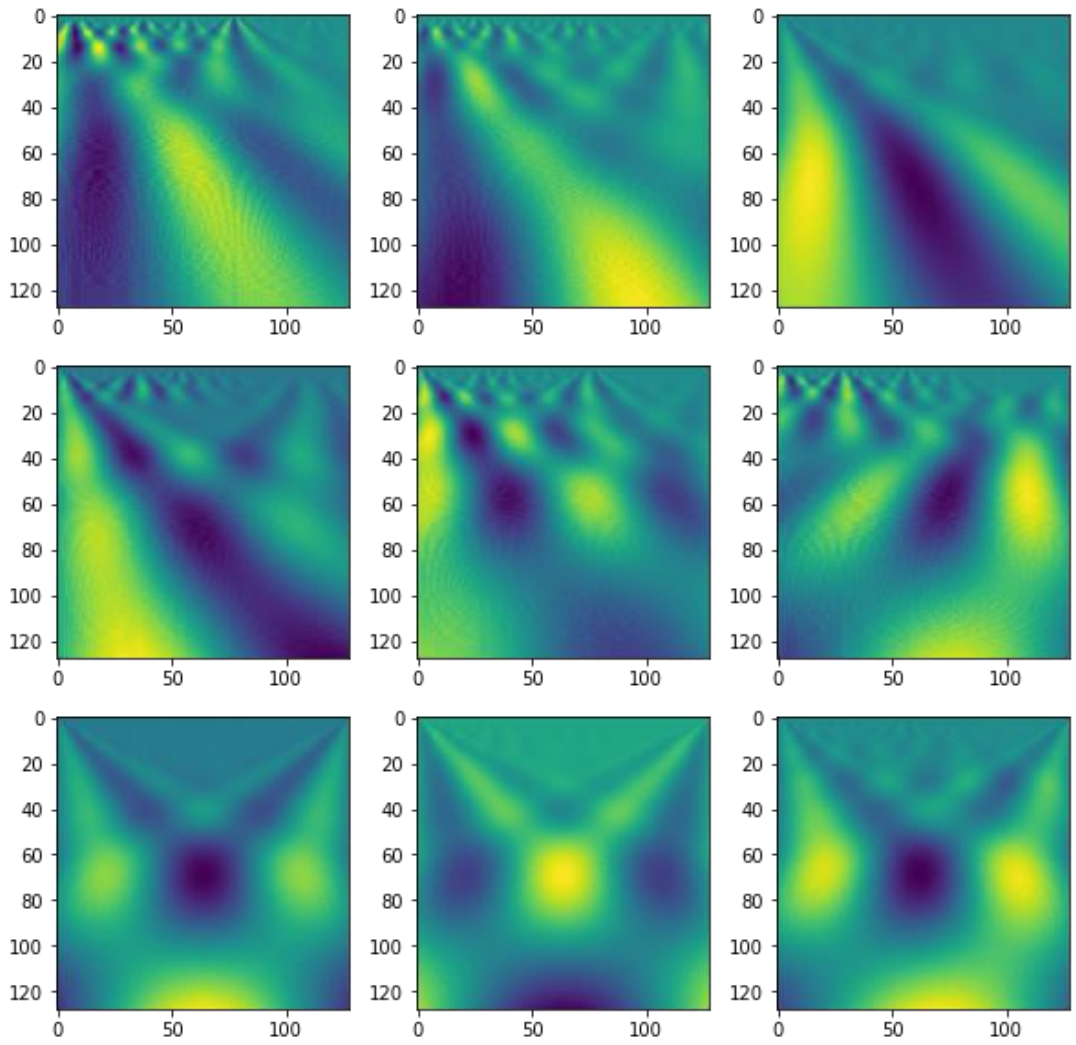
```
[ ] def create_cwt_images(X, n_scales, wavelet_name = "morl"):
        n_samples = X.shape[0]
        n_signals = X.shape[2]

        scales = np.arange(1,n_scales+1)

        X_cwt = np.ndarray(shape=(n_samples, n_scales, n_scales, n_signals), dtype = "float32")
        for sample in range(n_samples):
            if sample%100 == 0:
                print(sample)
            for signal in range(n_signals):
                serie=X[sample, :, signal]
                coeffs,freqs = pywt.cwt(serie,scales,wavelet_name)
                X_cwt[sample, :, :, signal]=coeffs

        return X_cwt

    n_scales=128
    X_test_cwt = create_cwt_images(x_test, n_scales)
```



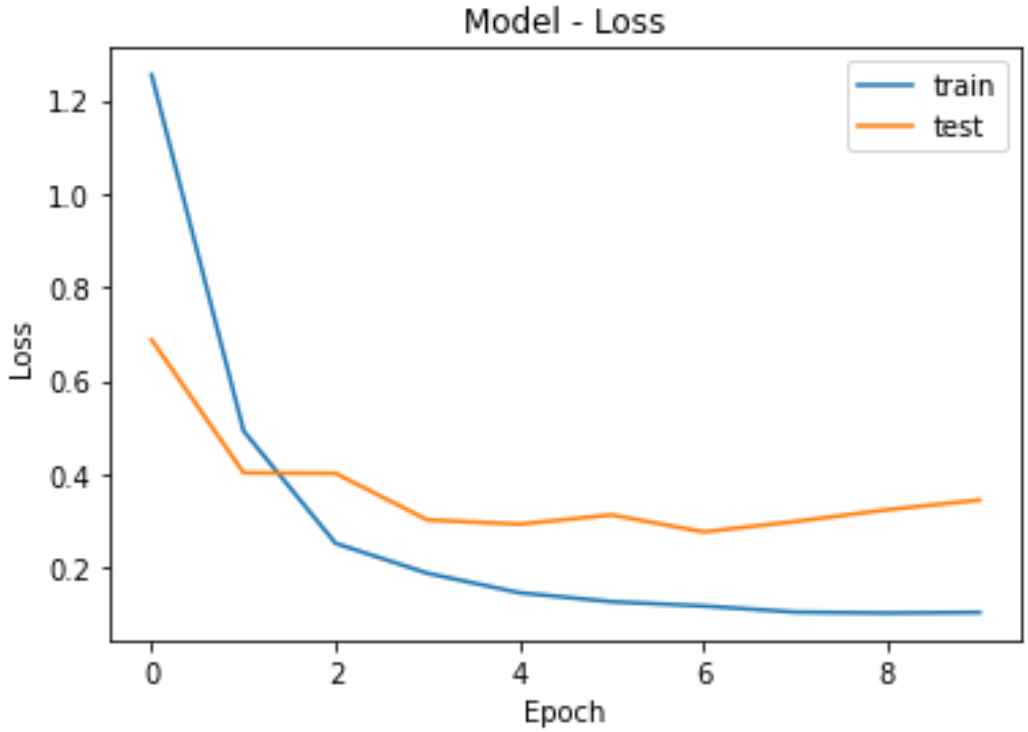
Comment: Every signal was transformed using the pywt.cwt function, with parameter wavelet set to "morl" – morlet wavelet, resulting in images similar to the ones shown to us during the lecture. Transformation of over 10000 signals took majority of computing time spent on this task.

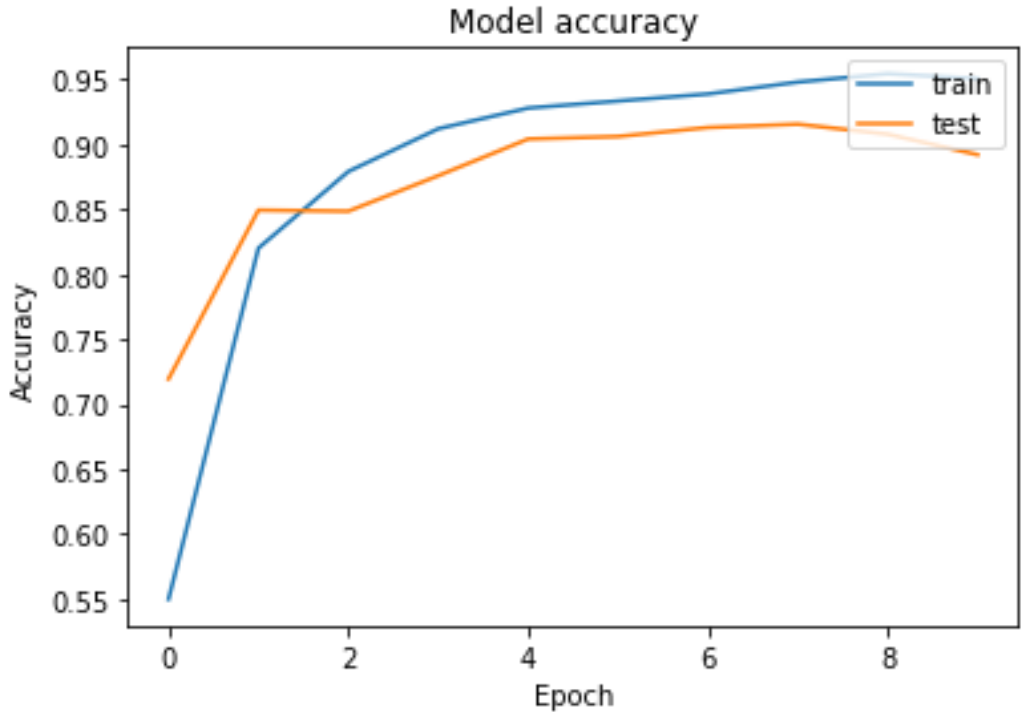
Task 3. Use keras library to build an CNN made up of two convolutional and two max pooling layers. Set kernel size to 5 by 5 and employ ReLU activation function. Use 10 epochs to train model.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 5)       1130
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 5)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 5)         630
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 5)         0
_________________________________________________________________
flatten (Flatten)            (None, 5120)              0
_________________________________________________________________
dense (Dense)                (None, 128)               655488
_________________________________________________________________
dense_1 (Dense)              (None, 54)                6966
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 330
=================================================================
Total params: 664,544
Trainable params: 664,544
Non-trainable params: 0
```

Comment: Training an entire model with given parameters took about 20 minutes.

Task 4. Plot training and test accuracies for each epoch





Comment: Model achieved accuracy close to 90%, before it started to begin overfitting.
9/10 times model will be able to predict correct action taken by device's user.
With increased filter size, or more densely populated Neural Network layer, it may be possible to achieve better results, but tendency towards overfitting, which can be noticed around 9th epoch shows us that we should be careful with such assumptions.