```python
import keras
from keras.losses import mean_squared_error as mse
from keras.optimizers import SGD

from mlair.model_modules import AbstractModelClass

from mlair.workflows import DefaultWorkflow

class MyCustomisedModel(AbstractModelClass):

    """
    A customised model with a 1x1 Conv, and 2 Dense layers (16,
    output shape). Dropout is used after Conv layer.
    """
    def __init__(self, input_shape: list, output_shape: list):

        # set attributes _input_shape and _output_shape
        super().__init__(input_shape[0], output_shape[0])

        # apply to model
        self.set_model()
        self.set_compile_options()
        self.set_custom_objects(loss=self.compile_options['loss'])

    def set_model(self):
        x_input = keras.layers.Input(shape=self._input_shape)
        x_in = keras.layers.Conv2D(32, (1, 1))(x_input)
        x_in = keras.layers.PReLU()(x_in)
        x_in = keras.layers.Flatten()(x_in)
        x_in = keras.layers.Dropout(0.1)(x_in)
        x_in = keras.layers.Dense(16)(x_in)
        x_in = keras.layers.PReLU()(x_in)
        x_in = keras.layers.Dense(self._output_shape)(x_in)
        out = keras.layers.PReLU()(x_in)
        self.model = keras.Model(inputs=x_input, outputs=[out])

    def set_compile_options(self):
        self.initial_lr = 1e-2
        self.optimizer = SGD(lr=self.initial_lr, momentum=0.9)
        self.loss = mse
        self.compile_options = {"metrics": ["mse", "mae"]}

# Make use of MyCustomisedModel within the DefaultWorkflow
workflow = DefaultWorkflow(model=MyCustomisedModel, epochs=2)
workflow.run()
```