*7. Zaproponuj własną architekturę głębokiego Autoencodera wykorzystującego filtry konwolucyjne. Nowe podejście do ekstrakcji cech powinno poprawić dokładność klasyfikacji na wszystkich zbiorach danych.

DATA:

```
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, recall_score, f1_score,
precision_score

(x_train, y_train), (x_test, y_test) =
tf.keras.datasets.mnist.load_data()
(f_x_train, f_y_train), (f_x_test, f_y_test) =
tf.keras.datasets.fashion_mnist.load_data()
k_x_train = np.load("kmnist-train-imgs.npz")['arr_0']
k_y_train = np.load("kmnist-train-labels.npz")['arr_0']
k_x_test = np.load("kmnist-test-imgs.npz")['arr_0']
k_y_test = np.load("kmnist-test-labels.npz")['arr_0']

y_train = np.array(y_train).astype(np.uint8)
y_test = np.array(y_test).astype(np.uint8)

f_y_train = np.array(f_y_train).astype(np.uint8)
f_y_test = np.array(f_y_test).astype(np.uint8)

k_y_train = np.array(k_y_train).astype(np.uint8)
k_y_test = np.array(k_y_test).astype(np.uint8)

x_train = x_train.astype(np.float32)/255.0
x_train = x_train.reshape(-1, 28, 28, 1)
f_x_train = f_x_train.astype(np.float32)/255.0
f_x_train = f_x_train.reshape(-1, 28, 28, 1)
k_x_train = k_x_train.astype(np.float32)/255.0
k_x_train = k_x_train.reshape(-1, 28, 28, 1)

x_test = x_test.astype(np.float32) / 255.0
x_test = x_test.reshape(-1, 28, 28, 1)
```

```
f_x_test = f_x_test.astype(np.float32) / 255.0
f_x_test = f_x_test.reshape(-1, 28, 28, 1)
k_x_test = k_x_test.astype(np.float32) / 255.0
k_x_test = k_x_test.reshape(-1, 28, 28, 1)

print(x_train.shape)
print(x_test.shape)
print(f_x_train.shape)
print(f_x_test.shape)
print(k_x_train.shape)
print(k_x_test.shape)
```

```
(60000, 28, 28, 1)
(10000, 28, 28, 1)
(60000, 28, 28, 1)
(10000, 28, 28, 1)
(60000, 28, 28, 1)
(10000, 28, 28, 1)
```

Autoencoder:

```
class AutoencoderAdvanced(tf.keras.Model):
  def __init__(self, encoding_dimension, input_shape):
    super(AutoencoderAdvanced, self).__init__()
    self.encoding_dimension = encoding_dimension
    self.dropout_rate = 0.2
    self.leaky_rate = 0.1
    self.encoder = tf.keras.Sequential(
        [
            tf.keras.layers.InputLayer(shape=input_shape),

            tf.keras.layers.Conv2D(filters=32, kernel_size=3,
strides=(2, 2)),
            tf.keras.layers.BatchNormalization(),
            tf.keras.layers.Dropout(self.dropout_rate),
            tf.keras.layers.LeakyReLU(negative_slope=self.leaky_rate),

            tf.keras.layers.Conv2D(filters=64, kernel_size=3,
strides=(2, 2)),
            tf.keras.layers.BatchNormalization(),
            tf.keras.layers.Dropout(self.dropout_rate),
            tf.keras.layers.LeakyReLU(negative_slope=self.leaky_rate),

            tf.keras.layers.Conv2D(filters=128, kernel_size=3,
strides=(2, 2)),
            tf.keras.layers.BatchNormalization(),
            tf.keras.layers.Dropout(self.dropout_rate),
            tf.keras.layers.LeakyReLU(negative_slope=self.leaky_rate),
```

```
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(512, activation='leaky_relu'),
            tf.keras.layers.Dropout(self.dropout_rate),
            tf.keras.layers.Dense(256, activation='leaky_relu'),
            tf.keras.layers.Dropout(self.dropout_rate),
            tf.keras.layers.Dense(self.encoding_dimension),
        ]
    )
    self.decoder = tf.keras.Sequential(
        [

tf.keras.layers.InputLayer(shape=(self.encoding_dimension,)),

            tf.keras.layers.Dense(256, activation='leaky_relu'),
            tf.keras.layers.Dropout(self.dropout_rate),
            tf.keras.layers.Dense(512, activation='leaky_relu'),
            tf.keras.layers.Dropout(self.dropout_rate),

            tf.keras.layers.Dense(units=7*7*32,
activation='leaky_relu'),
            tf.keras.layers.Reshape(target_shape=(7, 7, 32)),


            tf.keras.layers.Conv2DTranspose(filters=128,
kernel_size=3, strides=2, padding='same'),
            tf.keras.layers.BatchNormalization(),
            tf.keras.layers.Dropout(self.dropout_rate),
            tf.keras.layers.LeakyReLU(negative_slope=self.leaky_rate),

            tf.keras.layers.Conv2DTranspose(filters=64, kernel_size=3,
strides=2, padding='same'),
            tf.keras.layers.BatchNormalization(),
            tf.keras.layers.Dropout(self.dropout_rate),
            tf.keras.layers.LeakyReLU(negative_slope=self.leaky_rate),

            tf.keras.layers.Conv2DTranspose(filters=32, kernel_size=3,
strides=1, padding='same'),
            tf.keras.layers.BatchNormalization(),
            tf.keras.layers.Dropout(self.dropout_rate),
            tf.keras.layers.LeakyReLU(negative_slope=self.leaky_rate),

            tf.keras.layers.Conv2DTranspose(filters=1, kernel_size=3,
strides=1, padding='same', activation='sigmoid'),
        ]
    )

  def call(self, x):
    encoded = self.encoder(x)
    decoded = self.decoder(encoded)
    return decoded
```

Computations:

```
input_shape = (28, 28, 1)
latent_dim = 196
batch_size = 32
epochs = 40

autoencoder_adv = AutoencoderAdvanced(latent_dim, input_shape)
autoencoder_adv.compile(optimizer='adam',
loss=tf.keras.losses.BinaryCrossentropy(from_logits=False),
metrics=[tf.keras.metrics.Accuracy])

f_autoencoder_adv = AutoencoderAdvanced(latent_dim, input_shape)
f_autoencoder_adv.compile(optimizer='adam',
loss=tf.keras.losses.BinaryCrossentropy(from_logits=False),
metrics=[tf.keras.metrics.Accuracy])

k_autoencoder_adv = AutoencoderAdvanced(latent_dim, input_shape)
k_autoencoder_adv.compile(optimizer='adam',
loss=tf.keras.losses.BinaryCrossentropy(from_logits=False),
metrics=[tf.keras.metrics.Accuracy])

autoencoder_adv.fit(x_train, x_train,
                epochs=epochs,
                shuffle=True,
                batch_size=batch_size,
                validation_data=(x_test, x_test))
```

```
Epoch 1/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 141s 73ms/step - accuracy: 7.5776e-09 -
loss: 0.1670 - val_accuracy: 1.2755e-07 - val_loss: 0.0866
Epoch 2/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 131s 70ms/step - accuracy: 2.3359e-08 -
loss: 0.0863 - val_accuracy: 0.0000e+00 - val_loss: 0.0826
Epoch 3/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 117s 62ms/step - accuracy: 3.2392e-07 -
loss: 0.0813 - val_accuracy: 7.6531e-07 - val_loss: 0.0812
Epoch 4/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 4.6676e-06 -
loss: 0.0793 - val_accuracy: 1.9260e-05 - val_loss: 0.0786
Epoch 5/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 62ms/step - accuracy: 5.6172e-05 -
loss: 0.0778 - val_accuracy: 7.9209e-05 - val_loss: 0.0774
Epoch 6/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 62ms/step - accuracy: 1.5759e-04 -
loss: 0.0765 - val_accuracy: 1.7258e-04 - val_loss: 0.0766
Epoch 7/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 3.7182e-04 -
loss: 0.0755 - val_accuracy: 9.1416e-04 - val_loss: 0.0757
Epoch 8/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 9.5967e-04 -
```

```
loss: 0.0749 - val_accuracy: 9.4005e-04 - val_loss: 0.0752
Epoch 9/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 62ms/step - accuracy: 0.0013 -
loss: 0.0742 - val_accuracy: 0.0011 - val_loss: 0.0748
Epoch 10/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 0.0019 -
loss: 0.0737 - val_accuracy: 9.4184e-04 - val_loss: 0.0747
Epoch 11/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 59ms/step - accuracy: 0.0022 -
loss: 0.0731 - val_accuracy: 0.0029 - val_loss: 0.0739
Epoch 12/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 0.0029 -
loss: 0.0728 - val_accuracy: 0.0034 - val_loss: 0.0740
Epoch 13/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 0.0039 -
loss: 0.0724 - val_accuracy: 0.0041 - val_loss: 0.0736
Epoch 14/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 59ms/step - accuracy: 0.0047 -
loss: 0.0720 - val_accuracy: 0.0071 - val_loss: 0.0732
Epoch 15/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 117s 62ms/step - accuracy: 0.0053 -
loss: 0.0717 - val_accuracy: 0.0037 - val_loss: 0.0731
Epoch 16/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 122s 65ms/step - accuracy: 0.0058 -
loss: 0.0713 - val_accuracy: 0.0073 - val_loss: 0.0730
Epoch 17/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 119s 64ms/step - accuracy: 0.0078 -
loss: 0.0710 - val_accuracy: 0.0073 - val_loss: 0.0728
Epoch 18/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 0.0085 -
loss: 0.0707 - val_accuracy: 0.0097 - val_loss: 0.0724
Epoch 19/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 0.0098 -
loss: 0.0703 - val_accuracy: 0.0104 - val_loss: 0.0722
Epoch 20/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 60ms/step - accuracy: 0.0103 -
loss: 0.0702 - val_accuracy: 0.0114 - val_loss: 0.0719
Epoch 21/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 60ms/step - accuracy: 0.0119 -
loss: 0.0701 - val_accuracy: 0.0111 - val_loss: 0.0718
Epoch 22/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 60ms/step - accuracy: 0.0120 -
loss: 0.0698 - val_accuracy: 0.0152 - val_loss: 0.0720
Epoch 23/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 117s 62ms/step - accuracy: 0.0128 -
loss: 0.0697 - val_accuracy: 0.0084 - val_loss: 0.0718
Epoch 24/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 59ms/step - accuracy: 0.0135 -
loss: 0.0693 - val_accuracy: 0.0143 - val_loss: 0.0713
```

```
Epoch 25/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 60ms/step - accuracy: 0.0146 -
loss: 0.0692 - val_accuracy: 0.0151 - val_loss: 0.0714
Epoch 26/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 59ms/step - accuracy: 0.0145 -
loss: 0.0692 - val_accuracy: 0.0178 - val_loss: 0.0713
Epoch 27/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 59ms/step - accuracy: 0.0147 -
loss: 0.0690 - val_accuracy: 0.0138 - val_loss: 0.0713
Epoch 28/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 59ms/step - accuracy: 0.0149 -
loss: 0.0688 - val_accuracy: 0.0155 - val_loss: 0.0711
Epoch 29/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 0.0149 -
loss: 0.0687 - val_accuracy: 0.0156 - val_loss: 0.0709
Epoch 30/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 59ms/step - accuracy: 0.0158 -
loss: 0.0685 - val_accuracy: 0.0161 - val_loss: 0.0714
Epoch 31/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 60ms/step - accuracy: 0.0168 -
loss: 0.0685 - val_accuracy: 0.0191 - val_loss: 0.0711
Epoch 32/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 59ms/step - accuracy: 0.0170 -
loss: 0.0684 - val_accuracy: 0.0182 - val_loss: 0.0709
Epoch 33/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 60ms/step - accuracy: 0.0184 -
loss: 0.0681 - val_accuracy: 0.0191 - val_loss: 0.0710
Epoch 34/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 60ms/step - accuracy: 0.0179 -
loss: 0.0681 - val_accuracy: 0.0145 - val_loss: 0.0710
Epoch 35/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 60ms/step - accuracy: 0.0189 -
loss: 0.0680 - val_accuracy: 0.0170 - val_loss: 0.0709
Epoch 36/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 60ms/step - accuracy: 0.0184 -
loss: 0.0679 - val_accuracy: 0.0169 - val_loss: 0.0708
Epoch 37/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 60ms/step - accuracy: 0.0185 -
loss: 0.0677 - val_accuracy: 0.0211 - val_loss: 0.0710
Epoch 38/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 60ms/step - accuracy: 0.0192 -
loss: 0.0677 - val_accuracy: 0.0182 - val_loss: 0.0708
Epoch 39/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 60ms/step - accuracy: 0.0201 -
loss: 0.0676 - val_accuracy: 0.0233 - val_loss: 0.0706
Epoch 40/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 117s 62ms/step - accuracy: 0.0200 -
loss: 0.0675 - val_accuracy: 0.0212 - val_loss: 0.0709

<keras.src.callbacks.history.History at 0x7f946834e0d0>
```

```
f_autoencoder_adv.fit(f_x_train, f_x_train,
                epochs=epochs,
                shuffle=True,
                batch_size=batch_size,
                validation_data=(f_x_test, f_x_test))
```

```
Epoch 1/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 123s 64ms/step - accuracy: 1.7779e-07 -
loss: 0.3443 - val_accuracy: 0.0000e+00 - val_loss: 0.2941
Epoch 2/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 113s 60ms/step - accuracy: 1.4404e-07 -
loss: 0.2893 - val_accuracy: 1.2755e-07 - val_loss: 0.2861
Epoch 3/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 113s 60ms/step - accuracy: 6.4162e-08 -
loss: 0.2828 - val_accuracy: 0.0000e+00 - val_loss: 0.2825
Epoch 4/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 60ms/step - accuracy: 1.2064e-07 -
loss: 0.2797 - val_accuracy: 2.5510e-07 - val_loss: 0.2805
Epoch 5/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 114s 61ms/step - accuracy: 9.4011e-08 -
loss: 0.2780 - val_accuracy: 1.2755e-07 - val_loss: 0.2804
Epoch 6/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 113s 61ms/step - accuracy: 1.8468e-07 -
loss: 0.2767 - val_accuracy: 2.5510e-07 - val_loss: 0.2781
Epoch 7/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 61ms/step - accuracy: 5.4094e-08 -
loss: 0.2747 - val_accuracy: 0.0000e+00 - val_loss: 0.2779
Epoch 8/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 115s 62ms/step - accuracy: 1.4519e-07 -
loss: 0.2741 - val_accuracy: 3.8265e-07 - val_loss: 0.2770
Epoch 9/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 147s 64ms/step - accuracy: 6.4815e-08 -
loss: 0.2728 - val_accuracy: 1.2755e-07 - val_loss: 0.2764
Epoch 10/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 114s 61ms/step - accuracy: 1.9599e-07 -
loss: 0.2716 - val_accuracy: 0.0000e+00 - val_loss: 0.2763
Epoch 11/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 114s 61ms/step - accuracy: 2.8351e-08 -
loss: 0.2712 - val_accuracy: 0.0000e+00 - val_loss: 0.2760
Epoch 12/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 115s 61ms/step - accuracy: 1.7068e-07 -
loss: 0.2709 - val_accuracy: 1.2755e-07 - val_loss: 0.2746
Epoch 13/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 119s 64ms/step - accuracy: 2.2269e-07 -
loss: 0.2700 - val_accuracy: 1.2755e-07 - val_loss: 0.2743
Epoch 14/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 114s 61ms/step - accuracy: 2.6926e-07 -
loss: 0.2697 - val_accuracy: 1.2755e-07 - val_loss: 0.2740
Epoch 15/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 61ms/step - accuracy: 2.0696e-07 -
```

```
loss: 0.2690 - val_accuracy: 2.5510e-07 - val_loss: 0.2738
Epoch 16/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 144s 62ms/step - accuracy: 1.5315e-07 -
loss: 0.2692 - val_accuracy: 1.2755e-07 - val_loss: 0.2734
Epoch 17/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 118s 63ms/step - accuracy: 1.2290e-07 -
loss: 0.2688 - val_accuracy: 2.5510e-07 - val_loss: 0.2732
Epoch 18/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 115s 61ms/step - accuracy: 4.8663e-09 -
loss: 0.2677 - val_accuracy: 0.0000e+00 - val_loss: 0.2735
Epoch 19/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 61ms/step - accuracy: 2.0508e-07 -
loss: 0.2674 - val_accuracy: 0.0000e+00 - val_loss: 0.2731
Epoch 20/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 61ms/step - accuracy: 1.6989e-07 -
loss: 0.2675 - val_accuracy: 1.2755e-07 - val_loss: 0.2736
Epoch 21/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 61ms/step - accuracy: 2.7164e-07 -
loss: 0.2672 - val_accuracy: 0.0000e+00 - val_loss: 0.2731
Epoch 22/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 61ms/step - accuracy: 9.5375e-08 -
loss: 0.2662 - val_accuracy: 0.0000e+00 - val_loss: 0.2729
Epoch 23/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 61ms/step - accuracy: 2.1945e-07 -
loss: 0.2664 - val_accuracy: 0.0000e+00 - val_loss: 0.2726
Epoch 24/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 115s 61ms/step - accuracy: 1.8083e-07 -
loss: 0.2657 - val_accuracy: 0.0000e+00 - val_loss: 0.2744
Epoch 25/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 62ms/step - accuracy: 3.3574e-07 -
loss: 0.2666 - val_accuracy: 1.2755e-07 - val_loss: 0.2721
Epoch 26/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 62ms/step - accuracy: 1.4495e-07 -
loss: 0.2657 - val_accuracy: 1.2755e-07 - val_loss: 0.2722
Epoch 27/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 62ms/step - accuracy: 1.6093e-07 -
loss: 0.2651 - val_accuracy: 1.2755e-07 - val_loss: 0.2722
Epoch 28/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 121s 64ms/step - accuracy: 2.2742e-07 -
loss: 0.2648 - val_accuracy: 1.2755e-07 - val_loss: 0.2726
Epoch 29/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 137s 62ms/step - accuracy: 3.2854e-07 -
loss: 0.2645 - val_accuracy: 2.5510e-07 - val_loss: 0.2721
Epoch 30/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 62ms/step - accuracy: 5.6431e-08 -
loss: 0.2655 - val_accuracy: 2.5510e-07 - val_loss: 0.2723
Epoch 31/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 62ms/step - accuracy: 2.7392e-07 -
loss: 0.2645 - val_accuracy: 2.5510e-07 - val_loss: 0.2721
```

```
Epoch 32/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 115s 61ms/step - accuracy: 1.9687e-07 -
loss: 0.2636 - val_accuracy: 2.5510e-07 - val_loss: 0.2722
Epoch 33/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 121s 64ms/step - accuracy: 3.9863e-07 -
loss: 0.2637 - val_accuracy: 2.5510e-07 - val_loss: 0.2722
Epoch 34/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 62ms/step - accuracy: 2.3023e-07 -
loss: 0.2633 - val_accuracy: 0.0000e+00 - val_loss: 0.2721
Epoch 35/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 62ms/step - accuracy: 1.2324e-07 -
loss: 0.2629 - val_accuracy: 3.8265e-07 - val_loss: 0.2724
Epoch 36/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 120s 64ms/step - accuracy: 4.0508e-07 -
loss: 0.2636 - val_accuracy: 0.0000e+00 - val_loss: 0.2727
Epoch 37/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 62ms/step - accuracy: 4.9790e-07 -
loss: 0.2630 - val_accuracy: 2.4235e-06 - val_loss: 0.2723
Epoch 38/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 62ms/step - accuracy: 5.5375e-07 -
loss: 0.2630 - val_accuracy: 0.0000e+00 - val_loss: 0.2720
Epoch 39/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 62ms/step - accuracy: 2.0124e-07 -
loss: 0.2630 - val_accuracy: 1.2755e-07 - val_loss: 0.2721
Epoch 40/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 62ms/step - accuracy: 6.2502e-07 -
loss: 0.2628 - val_accuracy: 2.5510e-07 - val_loss: 0.2722

<keras.src.callbacks.history.History at 0x7f94670085d0>

k_autoencoder_adv.fit(k_x_train, k_x_train,
                epochs=epochs,
                shuffle=True,
                batch_size=batch_size,
                validation_data=(k_x_test, k_x_test))

Epoch 1/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 60ms/step - accuracy: 0.0000e+00 -
loss: 0.3119 - val_accuracy: 1.2755e-07 - val_loss: 0.2176
Epoch 2/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 0.0000e+00 -
loss: 0.1984 - val_accuracy: 0.0000e+00 - val_loss: 0.2023
Epoch 3/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 1.1416e-08 -
loss: 0.1840 - val_accuracy: 0.0000e+00 - val_loss: 0.1937
Epoch 4/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 0.0000e+00 -
loss: 0.1766 - val_accuracy: 0.0000e+00 - val_loss: 0.1887
Epoch 5/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 62ms/step - accuracy: 2.0016e-08 -
```

```
loss: 0.1725 - val_accuracy: 1.2755e-07 - val_loss: 0.1849
Epoch 6/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 5.2916e-08 -
loss: 0.1688 - val_accuracy: 0.0000e+00 - val_loss: 0.1839
Epoch 7/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 3.1330e-08 -
loss: 0.1667 - val_accuracy: 0.0000e+00 - val_loss: 0.1809
Epoch 8/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 0.0000e+00 -
loss: 0.1644 - val_accuracy: 0.0000e+00 - val_loss: 0.1794
Epoch 9/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 0.0000e+00 -
loss: 0.1624 - val_accuracy: 0.0000e+00 - val_loss: 0.1790
Epoch 10/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 62ms/step - accuracy: 2.2231e-08 -
loss: 0.1608 - val_accuracy: 0.0000e+00 - val_loss: 0.1770
Epoch 11/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 62ms/step - accuracy: 7.3268e-08 -
loss: 0.1594 - val_accuracy: 0.0000e+00 - val_loss: 0.1770
Epoch 12/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 5.5407e-08 -
loss: 0.1582 - val_accuracy: 0.0000e+00 - val_loss: 0.1758
Epoch 13/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 59ms/step - accuracy: 1.8288e-08 -
loss: 0.1567 - val_accuracy: 0.0000e+00 - val_loss: 0.1752
Epoch 14/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 0.0000e+00 -
loss: 0.1555 - val_accuracy: 0.0000e+00 - val_loss: 0.1745
Epoch 15/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 6.4846e-09 -
loss: 0.1543 - val_accuracy: 0.0000e+00 - val_loss: 0.1747
Epoch 16/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 1.8195e-10 -
loss: 0.1533 - val_accuracy: 0.0000e+00 - val_loss: 0.1739
Epoch 17/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 1.8837e-08 -
loss: 0.1525 - val_accuracy: 0.0000e+00 - val_loss: 0.1737
Epoch 18/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 7.1608e-08 -
loss: 0.1517 - val_accuracy: 1.2755e-07 - val_loss: 0.1733
Epoch 19/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 59ms/step - accuracy: 4.1645e-09 -
loss: 0.1510 - val_accuracy: 0.0000e+00 - val_loss: 0.1732
Epoch 20/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 1.3606e-07 -
loss: 0.1502 - val_accuracy: 0.0000e+00 - val_loss: 0.1730
Epoch 21/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 6.2317e-09 -
loss: 0.1496 - val_accuracy: 0.0000e+00 - val_loss: 0.1724
```

```
Epoch 22/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 6.3229e-09 -
loss: 0.1489 - val_accuracy: 0.0000e+00 - val_loss: 0.1723
Epoch 23/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 60ms/step - accuracy: 2.5079e-08 -
loss: 0.1483 - val_accuracy: 0.0000e+00 - val_loss: 0.1716
Epoch 24/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 62ms/step - accuracy: 3.8778e-08 -
loss: 0.1476 - val_accuracy: 1.2755e-07 - val_loss: 0.1720
Epoch 25/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 6.7458e-09 -
loss: 0.1469 - val_accuracy: 0.0000e+00 - val_loss: 0.1717
Epoch 26/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 1.8735e-08 -
loss: 0.1464 - val_accuracy: 0.0000e+00 - val_loss: 0.1715
Epoch 27/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 59ms/step - accuracy: 0.0000e+00 -
loss: 0.1460 - val_accuracy: 0.0000e+00 - val_loss: 0.1719
Epoch 28/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 2.6604e-08 -
loss: 0.1452 - val_accuracy: 0.0000e+00 - val_loss: 0.1719
Epoch 29/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 1.1630e-08 -
loss: 0.1453 - val_accuracy: 1.2755e-07 - val_loss: 0.1715
Epoch 30/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 3.7013e-09 -
loss: 0.1445 - val_accuracy: 0.0000e+00 - val_loss: 0.1719
Epoch 31/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 2.9303e-08 -
loss: 0.1439 - val_accuracy: 0.0000e+00 - val_loss: 0.1721
Epoch 32/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 60ms/step - accuracy: 0.0000e+00 -
loss: 0.1437 - val_accuracy: 0.0000e+00 - val_loss: 0.1714
Epoch 33/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 142s 59ms/step - accuracy: 7.2262e-09 -
loss: 0.1432 - val_accuracy: 0.0000e+00 - val_loss: 0.1720
Epoch 34/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 116s 62ms/step - accuracy: 3.8145e-08 -
loss: 0.1430 - val_accuracy: 0.0000e+00 - val_loss: 0.1716
Epoch 35/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 2.1624e-10 -
loss: 0.1425 - val_accuracy: 0.0000e+00 - val_loss: 0.1721
Epoch 36/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 59ms/step - accuracy: 2.5533e-08 -
loss: 0.1422 - val_accuracy: 0.0000e+00 - val_loss: 0.1724
Epoch 37/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 111s 59ms/step - accuracy: 3.7172e-08 -
loss: 0.1420 - val_accuracy: 0.0000e+00 - val_loss: 0.1714
Epoch 38/40
```

```
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 147s 62ms/step - accuracy: 2.1693e-08 -
loss: 0.1416 - val_accuracy: 0.0000e+00 - val_loss: 0.1721
Epoch 39/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 60ms/step - accuracy: 1.3714e-08 -
loss: 0.1413 - val_accuracy: 1.2755e-07 - val_loss: 0.1732
Epoch 40/40
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 112s 59ms/step - accuracy: 0.0000e+00 -
loss: 0.1409 - val_accuracy: 0.0000e+00 - val_loss: 0.1720

<keras.src.callbacks.history.History at 0x7f9466593fd0>

reconstructions_x_train = autoencoder_adv.predict(x_train)
reconstructions_x_train = reconstructions_x_train.reshape(-1, 28 * 28)
reconstructions_x_test = autoencoder_adv.predict(x_test)
reconstructions_x_test = reconstructions_x_test.reshape(-1, 28 * 28)

reconstructions_f_x_train = f_autoencoder_adv.predict(f_x_train)
reconstructions_f_x_train = reconstructions_f_x_train.reshape(-1, 28 *
28)
reconstructions_f_x_test = f_autoencoder_adv.predict(f_x_test)
reconstructions_f_x_test = reconstructions_f_x_test.reshape(-1, 28 *
28)

reconstructions_k_x_train = k_autoencoder_adv.predict(k_x_train)
reconstructions_k_x_train = reconstructions_k_x_train.reshape(-1, 28 *
28)
reconstructions_k_x_test = k_autoencoder_adv.predict(k_x_test)
reconstructions_k_x_test = reconstructions_k_x_test.reshape(-1, 28 *
28)

1875/1875 ━━━━━━━━━━━━━━━━━━━━ 31s 17ms/step
313/313 ━━━━━━━━━━━━━━━━━━━━ 5s 17ms/step
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 31s 17ms/step
313/313 ━━━━━━━━━━━━━━━━━━━━ 5s 17ms/step
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 32s 17ms/step
313/313 ━━━━━━━━━━━━━━━━━━━━ 5s 17ms/step
```

Classification:

```
x_train = x_train.reshape(-1, 784)
f_x_train = f_x_train.reshape(-1, 784)
k_x_train = k_x_train.reshape(-1, 784)

x_test = x_test.reshape(-1, 784)
f_x_test = f_x_test.reshape(-1, 784)
k_x_test = k_x_test.reshape(-1, 784)

classifier = LogisticRegression(solver='newton-cg')
classifier.fit(x_train, y_train)
```

```
y_pred = classifier.predict(x_test)
y_pred_recon = classifier.predict(reconstructions_x_test)
print(f"accuracy_score baseline: {accuracy_score(y_pred, y_test)},
accuracy_score reconstructions: {accuracy_score(y_pred_recon,
y_test)}")
```

```
accuracy_score baseline: 0.9265, accuracy_score reconstructions: 0.929
```

```
f_classifier = LogisticRegression(solver='newton-cg')
f_classifier.fit(f_x_train, f_y_train)

y_pred = f_classifier.predict(f_x_test)
y_pred_recon = f_classifier.predict(reconstructions_f_x_test)
print(f"accuracy_score baseline: {accuracy_score(y_pred, f_y_test)},
accuracy_score reconstructions: {accuracy_score(y_pred_recon,
f_y_test)}")
```

```
accuracy_score baseline: 0.8436, accuracy_score reconstructions:
0.8463
```

```
k_classifier = LogisticRegression(solver='newton-cg')
k_classifier.fit(k_x_train, k_y_train)

y_pred = k_classifier.predict(k_x_test)
y_pred_recon = k_classifier.predict(reconstructions_k_x_test)
print(f"accuracy_score baseline: {accuracy_score(y_pred, k_y_test)},
accuracy_score reconstructions: {accuracy_score(y_pred_recon,
k_y_test)}")
```

```
accuracy_score baseline: 0.6943, accuracy_score reconstructions:
0.7074
```

```
rf_classifier = RandomForestClassifier()
rf_classifier.fit(x_train, y_train)

y_pred_rf = rf_classifier.predict(x_test)
y_pred_rf_recon = rf_classifier.predict(reconstructions_x_test)
print(f"accuracy_score baseline: {accuracy_score(y_pred_rf, y_test)},
accuracy_score reconstructions: {accuracy_score(y_pred_rf_recon,
y_test)}")
```

```
accuracy_score baseline: 0.9683, accuracy_score reconstructions:
0.9654
```

```
f_rf_classifier = RandomForestClassifier()
f_rf_classifier.fit(f_x_train, f_y_train)
y_pred_rf = f_rf_classifier.predict(f_x_test)
y_pred_rf_recon = f_rf_classifier.predict(reconstructions_f_x_test)
print(f"accuracy_score baseline: {accuracy_score(y_pred_rf,
f_y_test)}, accuracy_score reconstructions:
{accuracy_score(y_pred_rf_recon, f_y_test)}")
```

```
accuracy_score baseline: 0.8777, accuracy_score reconstructions:
0.8482
```

```
k_rf_classifier = RandomForestClassifier()
k_rf_classifier.fit(k_x_train, k_y_train)
y_pred_rf = k_rf_classifier.predict(k_x_test)
y_pred_rf_recon = k_rf_classifier.predict(reconstructions_k_x_test)
print(f"accuracy_score baseline: {accuracy_score(y_pred_rf,
k_y_test)}, accuracy_score reconstructions:
{accuracy_score(y_pred_rf_recon, k_y_test)}")
```

```
accuracy_score baseline: 0.8538, accuracy_score reconstructions:
0.8226
```

Wnioski:

Wszystkie wyniki rekonstrukcji zostały poprawione względem autoencodera z poprzedniego zadania. Dla klasyfikatora logistic regression wyniki zostały poprawione po zastosowaniu ekstrakcji cech względem baseline-u. Dla klasyfikatora random forest nie udało się poprawić wyników względem baseline-u, ale są one lepsze niż dla logistic regression.