ETAP 1: Zamontowanie bibliotek i danych

In [1]:
```
import os, shutil, glob
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import pandas as pd
import tqdm
import keras
import tensorflow as tf
```

In [2]:
```
DATA_DIR = '/content/drive/MyDrive/data/'
train_dir = os.path.join(DATA_DIR, 'training_data')
test_dir = os.path.join(DATA_DIR, 'testing_data')
meta_dir = os.path.join(DATA_DIR, 'meta')
meta_csv = os.path.join(DATA_DIR, 'meta.csv')
meta = pd.read_csv(meta_csv, sep=';')
meta.shape
```

In [3]:
```
classes = sorted(os.listdir(train_dir))
classes = [c for c in classes if '.DS_Store' not in c]
classes = [c for c in classes if '.ipynb_checkpoints' not in c]
classes = [c for c in classes if '.DS_Store' not in c]
classes
```

In [4]:
```
len(classes)
```

Out[4]:
```
43
```

In [5]:
```
meta.shape
```

Out[5]:
```
(43, 3)
```

ETAP 2: Wizualizacja zdjęć

In [6]:
```
# ścieżka do losowego obrazka z klasy "00000"
sample_dir = os.path.join(train_dir, '00000')
sample = os.listdir(sample_dir)[0]
sample_path = os.path.join(sample_dir, sample)

# wczytanie i wyświetlenie
from PIL import Image
img = Image.open(sample_path)
plt.imshow(img)
plt.title("Przykładowy obrazek z klasy 00000")
plt.axis('off')
plt.show()
```

Out[6]:

In [7]:
```
# ustawienia
num_samples = 5
plt.figure(figure=(10, 5))

# pętla po klasach
for i, cls in enumerate(classes[:5]):
    cls_dir = os.path.join(train_dir, cls)
    files = os.listdir(cls_dir)[:num_samples]
    for j, file in enumerate(files):
        img = Image.open(os.path.join(cls_dir, file))
        plt.subplot(5, num_samples, i*num_samples + j + 1)
        plt.imshow(img)
        plt.axis('off')
        if j == 0:
            plt.title(cls)
plt.suptitle("Przykładowe obrazki z klas 00000–00004")
plt.tight_layout()
plt.show()
```

Out[7]:

ETAP 3: Konwersja obrazów na skalę szarości i normalizacja

In [8]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42, stratify=y)
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
```

In [9]:
```
# Tworzenie folderów wyjściowych
train_gray_dir = '/content/drive/MyDrive/data/training_data_gray'
test_gray_dir = '/content/drive/MyDrive/data/testing_data_gray'
os.makedirs(train_gray_dir, exist_ok=True)
os.makedirs(test_gray_dir, exist_ok=True)
```

In [10]:
```
X_train[0].shape
```

Out[10]:
```
(1360, 50, 50, 3)
```

In [11]:
```
# przekonwertuj pierwszy obrazek
from PIL import Image, ImageOps
img = Image.open(X_train[0])
gray = ImageOps.grayscale(img)
print("Oryginalny:", X_train[0])
print("Szarość:", gray.size, gray.mode)
```

In [12]:
```
X_train[0].shape
```

Out[12]:
```
(1360, 50, 50, 3)
```

In [13]:
```
# przekonwertuj wszystkie
for i, path in tqdm.tqdm(enumerate(X_train), total=len(X_train)):
    img = Image.open(path)
    gray = ImageOps.grayscale(img)
    gray.save(os.path.join(train_gray_dir, f"{i:5d}.jpg"))
```

Out[13]:
```
100%|██████████| 1360/1360
```

In [14]:
```
# przekonwertuj X_test
for i, path in tqdm.tqdm(enumerate(X_test), total=len(X_test)):
    img = Image.open(path)
    gray = ImageOps.grayscale(img)
    gray.save(os.path.join(test_gray_dir, f"{i:5d}.jpg"))
```

Out[14]:
```
100%|██████████| 340/340
```

ETAP 4: Zbudowanie głębokiej konwolucyjnej sieci neuronowej

In [15]:
```
from keras.models import Sequential, Input, Model, regularizers
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D, BatchNormalization, GlobalAveragePooling2D
from keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
from keras.utils.vis_utils import plot_model
from keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint
from keras.optimizers import Adam, SGD, RMSprop
from keras.utils.vis_utils import plot_model, pydot, graphviz
```

In [16]:
```
# wczytanie obrazów z folderów
train_ds = tf.keras.utils.image_dataset_from_directory(
    train_gray_dir,
    image_size=(50, 50),
    batch_size=32,
    color_mode='grayscale',
    shuffle=True,
    seed=42,
    validation_split=0.2,
    subset='training'
)

test_ds = tf.keras.utils.image_dataset_from_directory(
    test_gray_dir,
    image_size=(50, 50),
    batch_size=32,
    color_mode='grayscale',
    shuffle=False
)

# sprawdzenie klas
print("Klasy:", train_ds.class_names)
```

Out[16]:
```
Found 1360 files belonging to 43 classes.
Found 340 files belonging to 43 classes.
Klasy: ['00000', '00001', '00002', '00003', '00004', '00005', '00006', '00007', '00008', '00009', '00010', '00011', '00012', '00013', '00014', '00015', '00016', '00017', '00018', '00019', '00020', '00021', '00022', '00023', '00024', '00025', '00026', '00027', '00028', '00029', '00030', '00031', '00032', '00033', '00034', '00035', '00036', '00037', '00038', '00039', '00040', '00041', '00042']
```

In [17]:
```
# normalizacja pikseli do zakresu [0, 1]
train_ds = train_ds.map(lambda x, y: (x / 255.0, y))
test_ds = test_ds.map(lambda x, y: (x / 255.0, y))

# sprawdzenie kształtu batcha
for x, y in train_ds.take(1):
    print("Batch shape:", x.shape, y.shape)
```

Out[17]:
```
Batch shape: (32, 50, 50, 1) (32,)
```

In [18]:
```
# wyświetlenie 9 przykładowych obrazków
plt.figure(figure=(10, 10))
for x, y in train_ds.take(1):
    for i in range(9):
        plt.subplot(3, 3, i+1)
        plt.imshow(x[i].numpy(), cmap='gray')
        plt.title(train_ds.class_names[y[i]])
        plt.axis('off')
plt.suptitle("Przykładowe obrazki z treningowego")
plt.tight_layout()
plt.show()
```

Out[18]:

ETAP 5: Kompilowanie i trenowanie głębokiej konwolucyjnej sieci neuronowej

In [19]:
```
# Wyświetlenie modelu
model.summary()
```

Out[19]:
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 48, 48, 32)        320
 batch_normalization (BatchNormalization)  (None, 48, 48, 32)  128
 max_pooling2d (MaxPooling2D)  (None, 24, 24, 32)      0
 dropout (Dropout)           (None, 24, 24, 32)        0
 conv2d_1 (Conv2D)           (None, 22, 22, 64)        18496
 batch_normalization_1 (BatchNormalization)  (None, 22, 22, 64)  256
 max_pooling2d_1 (MaxPooling2D)  (None, 11, 11, 64)    0
 dropout_1 (Dropout)         (None, 11, 11, 64)        0
 conv2d_2 (Conv2D)           (None, 9, 9, 128)         73856
 batch_normalization_2 (BatchNormalization)  (None, 9, 9, 128)  512
 dropout_2 (Dropout)         (None, 9, 9, 128)         0
 flatten (Flatten)           (None, 10368)             0
 dense (Dense)               (None, 128)               1327488
 dropout_3 (Dropout)         (None, 128)               0
 dense_1 (Dense)             (None, 43)                5547
=================================================================
Total params: 1426659
Trainable params: 1426179
Non-trainable params: 480
_________________________________________________________________
```

In [20]:
```
# Kompilowanie modelu
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

In [21]:
```
# Trenowanie z EarlyStopping i ReduceLROnPlateau
history = model.fit(
    train_ds,
    validation_ds,
    epochs=50,
    callbacks=[
        EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True),
        ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, verbose=1)
    ]
)
```

Out[21]:
```
Epoch 1/50
43/43 [==============================] - ETA: 0s - loss: 3.5171 - accuracy: 0.0882
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
43/43 [==============================] - 2s 47ms/step - loss: 3.5171 - accuracy: 0.0882 - val_loss: 3.7670 - lr: 0.0010
Epoch 2/50
43/43 [==============================] - ETA: 0s - loss: 2.8744 - accuracy: 0.2147
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
43/43 [==============================] - 2s 44ms/step - loss: 2.8744 - accuracy: 0.2147 - val_loss: 3.8844 - lr: 0.0010
Epoch 3/50
43/43 [==============================] - ETA: 0s - loss: 2.6202 - accuracy: 0.3213
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
43/43 [==============================] - 2s 44ms/step - loss: 2.6202 - accuracy: 0.3213 - val_loss: 4.7761 - lr: 0.0010
Epoch 4/50
43/43 [==============================] - ETA: 0s - loss: 2.3967 - accuracy: 0.4037
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
43/43 [==============================] - 2s 44ms/step - loss: 2.3967 - accuracy: 0.4037 - val_loss: 5.0557 - lr: 0.0010
Epoch 5/50
43/43 [==============================] - ETA: 0s - loss: 2.2011 - accuracy: 0.4728
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
43/43 [==============================] - 2s 44ms/step - loss: 2.2011 - accuracy: 0.4728 - val_loss: 5.0000 - lr: 0.0010
Epoch 6/50
43/43 [==============================] - ETA: 0s - loss: 2.0000 - accuracy: 0.5515
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
43/43 [==============================] - 2s 44ms/step - loss: 2.0000 - accuracy: 0.5515 - val_loss: 5.0752 - lr: 0.0010
Epoch 7/50
43/43 [==============================] - ETA: 0s - loss: 1.8408 - accuracy: 0.5927
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
43/43 [==============================] - 2s 44ms/step - loss: 1.8408 - accuracy: 0.5927 - val_loss: 5.0462 - lr: 0.0010
Epoch 8/50
43/43 [==============================] - ETA: 0s - loss: 1.6842 - accuracy: 0.6544
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
43/43 [==============================] - 2s 44ms/step - loss: 1.6842 - accuracy: 0.6544 - val_loss: 5.4229 - lr: 0.0010
Epoch 9/50
43/43 [==============================] - ETA: 0s - loss: 1.5711 - accuracy: 0.6941
WARNING:tensorflow:Can save best model only with val_loss available, skipping.
43/43 [==============================] - 2s 44ms/step - loss: 1.5711 - accuracy: 0.6941 - val_loss: 5.0000 - lr: 0.0010
```

ETAP 6: Ocena działania sieci

In [22]:
```
# Wykres dokładności
plt.figure(figure=(8, 5))
plt.plot(history.history['accuracy'], label='Training accuracy')
plt.plot(history.history['val_accuracy'], label='Validation accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Out[22]:

In [23]:
```
# Wykres strat
plt.figure(figure=(8, 5))
plt.plot(history.history['loss'], label='Training loss')
plt.plot(history.history['val_loss'], label='Validation loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training and validation loss')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

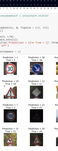
Out[23]:

In [24]:
```
# Macierz pomyłek
from sklearn.metrics import confusion_matrix, classification_report
y_true = []
y_pred = []
for x, y in test_ds:
    y_true.extend(y.numpy())
    y_pred.extend(model.predict(x).argmax(axis=1))
cm = confusion_matrix(y_true, y_pred)
print(cm.shape)
```

Out[24]:
```
43/43 [==============================] - 1s 13ms/step
(43, 43)
```

In [25]:
```
# Wizualizacja macierzy pomyłek
import seaborn as sns
plt.figure(figure=(12, 10))
sns.heatmap(cm, cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion matrix')
plt.tight_layout()
plt.show()
```

Out[25]:



In [26]:
```
# Raport klasyfikacji
print(classification_report(y_true, y_pred, zero_division=0))
```

Out[26]:
```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         8
           1       0.00      0.00      0.00         8
           2       0.00      0.00      0.00         8
           3       0.00      0.00      0.00         8
           4       0.00      0.00      0.00         8
           5       0.00      0.00      0.00         8
           6       0.00      0.00      0.00         8
           7       0.00      0.00      0.00         8
           8       0.00      0.00      0.00         8
           9       0.00      0.00      0.00         8
          10       0.00      0.00      0.00         8
          11       0.00      0.00      0.00         8
          12       0.00      0.00      0.00         8
          13       0.00      0.00      0.00         8
          14       0.00      0.00      0.00         8
          15       0.00      0.00      0.00         8
          16       0.00      0.00      0.00         8
          17       0.00      0.00      0.00         8
          18       0.00      0.00      0.00         8
          19       0.00      0.00      0.00         8
          20       0.00      0.00      0.00         8
          21       0.00      0.00      0.00         8
          22       0.00      0.00      0.00         8
          23       0.00      0.00      0.00         8
          24       0.00      0.00      0.00         8
          25       0.00      0.00      0.00         8
          26       0.00      0.00      0.00         8
          27       0.00      0.00      0.00         8
          28       0.00      0.00      0.00         8
          29       0.00      0.00      0.00         8
          30       0.00      0.00      0.00         8
          31       0.00      0.00      0.00         8
          32       0.00      0.00      0.00         8
          33       0.00      0.00      0.00         8
          34       0.00      0.00      0.00         8
          35       0.00      0.00      0.00         8
          36       0.00      0.00      0.00         8
          37       0.00      0.00      0.00         8
          38       0.00      0.00      0.00         8
          39       0.00      0.00      0.00         8
          40       0.00      0.00      0.00         8
          41       0.00      0.00      0.00         8
          42       0.00      0.00      0.00         8

    accuracy                           0.00       340
   macro avg       0.00      0.00      0.00       340
weighted avg       0.00      0.00      0.00       340
```

In [27]:
```
# Wyświetlenie 9 losowych obrazków z ich przewidzeniami
import random
plt.figure(figure=(10, 10))
for i in range(9):
    idx = random.randint(0, len(x_test))
    plt.subplot(3, 3, i+1)
    plt.imshow(x_test[idx].reshape(50, 50), cmap='gray')
    plt.title(f'Pred: {y_pred[idx]}\nTrue: {y_test[idx]}')
    plt.axis('off')
plt.suptitle('Przykładowe obrazki z ich przewidzeniami')
plt.tight_layout()
plt.show()
```

Out[27]: