SPRAWOZDANIE

Zajęcia: Nauka o danych II

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium Nr 2 Data 10.05.2025 Temat: „Projektowanie zaawansowanych architektur sieci neuronowych w TensorFlow lub PyTorch” Wariant 1	Szymon Nycz Informatyka II stopień, niestacjonarne, 2 semestr, gr.1a TTO

1. Polecenie:

   Link do repozytorium:
   https://github.com/Maciek332/Semestr_3_Nycz/tree/main/NoD%20II

   Zadanie 1 (U-Net): Zastosuj U-Net do segmentacji dróg w obrazach satelitarnych (DeepGlobe) — DeepGlobe Road Extraction Dataset. Oto link
   Dane: Obrazy RGB (1024x1024) i maski binarne dróg. Przeskaluj do 256×256, zamień maski do formatu 0/1.

   Współczesne architektury sieci neuronowych, takie jak U-Net, Autoencoder, Encoder-Decoder, BRNN, GAN i Transformer, znacząco rozszerzyły możliwości głębokiego uczenia. U-Net sprawdza się w segmentacji obrazów dzięki symetrycznej budowie i połączeniom skip. Autoenkodery umożliwiają kompresję i rekonstrukcję danych. Architektury encoder-decoder i BRNN skutecznie analizują sekwencje dzięki wykorzystaniu kontekstu. GAN-y generują realistyczne dane, a Transformery, oparte na mechanizmie uwagi, oferują dużą skuteczność i równoległe przetwarzanie sekwencji.

2. Opis programu opracowanego

```python
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt

def conv_block(x, filters):
    x = layers.Conv2D(filters, 3, activation='relu', padding='same')(x)
    x = layers.Conv2D(filters, 3, activation='relu', padding='same')(x)
    return x

def encoder_block(x, filters):
    f = conv_block(x, filters)
    p = layers.MaxPooling2D((2, 2))(f)
    return f, p

def decoder_block(x, skip, filters):
    x = layers.Conv2DTranspose(filters, (2, 2), strides=2, padding='same')(x)
    x = layers.concatenate([x, skip])
    x = conv_block(x, filters)
    return x

def build_unet(input_shape):
    inputs = layers.Input(shape=input_shape)

    s1, p1 = encoder_block(inputs, 64)
    s2, p2 = encoder_block(p1, 128)
    s3, p3 = encoder_block(p2, 256)

    b1 = conv_block(p3, 512)

    d1 = decoder_block(b1, s3, 256)
    d2 = decoder_block(d1, s2, 128)
    d3 = decoder_block(d2, s1, 64)

    outputs = layers.Conv2D(1, 1, activation='sigmoid')(d3)

    model = models.Model(inputs, outputs)
    return model

# Inicjalizacja modelu
model = build_unet((256, 256, 3))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.summary()
```

✓ 23s

Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 256, 256, 3)	0	-
conv2d (Conv2D)	(None, 256, 256, 64)	1,792	input_layer[0][0]
conv2d_1 (Conv2D)	(None, 256, 256, 64)	36,928	conv2d[0][0]
max_pooling2d (MaxPooling2D)	(None, 128, 128, 64)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 128, 128, 128)	73,856	max_pooling2d[0]…
conv2d_3 (Conv2D)	(None, 128, 128, 128)	147,584	conv2d_2[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 128)	0	conv2d_3[0][0]
conv2d_4 (Conv2D)	(None, 64, 64, 256)	295,168	max_pooling2d_1[…
conv2d_5 (Conv2D)	(None, 64, 64, 256)	590,080	conv2d_4[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 32, 32, 256)	0	conv2d_5[0][0]
conv2d_6 (Conv2D)	(None, 32, 32, 512)	1,180,160	max_pooling2d_2[…
conv2d_7 (Conv2D)	(None, 32, 32, 512)	2,359,808	conv2d_6[0][0]
conv2d_transpose (Conv2DTranspose)	(None, 64, 64, 256)	524,544	conv2d_7[0][0]
concatenate (Concatenate)	(None, 64, 64, 512)	0	conv2d_transpose… conv2d_5[0][0]
concatenate (Concatenate)	(None, 64, 64, 512)	0	conv2d_transpose… conv2d_5[0][0]
conv2d_8 (Conv2D)	(None, 64, 64, 256)	1,179,904	concatenate[0][0]
conv2d_9 (Conv2D)	(None, 64, 64, 256)	590,080	conv2d_8[0][0]
conv2d_transpose_1 (Conv2DTranspose)	(None, 128, 128, 128)	131,200	conv2d_9[0][0]
concatenate_1 (Concatenate)	(None, 128, 128, 256)	0	conv2d_transpose… conv2d_3[0][0]
conv2d_10 (Conv2D)	(None, 128, 128, 128)	295,040	concatenate_1[0]…
conv2d_11 (Conv2D)	(None, 128, 128, 128)	147,584	conv2d_10[0][0]
conv2d_transpose_2 (Conv2DTranspose)	(None, 256, 256, 64)	32,832	conv2d_11[0][0]
concatenate_2 (Concatenate)	(None, 256, 256, 128)	0	conv2d_transpose… conv2d_1[0][0]
conv2d_12 (Conv2D)	(None, 256, 256, 64)	73,792	concatenate_2[0]…
conv2d_13 (Conv2D)	(None, 256, 256, 64)	36,928	conv2d_12[0][0]
conv2d_14 (Conv2D)	(None, 256, 256, 1)	65	conv2d_13[0][0]

Total params: 7,697,345 (29.36 MB)

Trainable params: 7,697,345 (29.36 MB)

Non-trainable params: 0 (0.00 B)

3. Wnioski

Ćwiczenie umożliwiło praktyczne poznanie zaawansowanych architektur sieci neuronowych oraz ich zastosowań. Studenci zrozumieli, jak dobór architektury wpływa na skuteczność modelu w różnych zadaniach, takich jak segmentacja, analiza sekwencji czy generacja danych. Zdobyta wiedza stanowi podstawę do samodzielnego projektowania nowoczesnych modeli głębokiego uczenia.