deeplearningbeadandofinished

December 21, 2023

1 Beadandó feladat!

1.1 Deep Learning 2003 beadandó feladat!

Készítsen az alábbiakban letölthető fájlok segítségével egy a röntgen felvételeket értékelő bináris osztályozót. A felvételeken egészséges és beteg emberekhez tartozó röntgen képek találhatóak. A feladatot konvolúciós neurális hálózat segítségével valósítsa meg! Használjon korai leállítást, értékelje a modellt annak a pontosságával! Rajzolja ki a költségfügvény alakaulását (Loss) és szövegesen értékelje ki! Az elkészült modellt mentse el. A test adatokon értékelje a modelljét! A legjobb eredményt folyamatosan megosztom a Teamsben.

Beküldési határidő: 2023.december 22.

1.1.1 Legeslegelőször beinportáljuk a szükséges könyvtárakat.

[1]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf
from tensorflow.keras import layers, applications
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, CSVLogger
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model
import gdown
import zipfile
import os
```

```
2023-12-21 18:21:56.174195: E
external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register
cuDNN factory: Attempting to register factory for plugin cuDNN when one has
already been registered
2023-12-21 18:21:56.174227: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register
cuFFT factory: Attempting to register factory for plugin cuFFT when one has
already been registered
2023-12-21 18:21:56.175436: E
external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to
register cuBLAS factory: Attempting to register factory for plugin cuBLAS when
one has already been registered
```

```
2023-12-21 18:21:56.182229: I tensorflow/core/platform/cpu_feature_guard.cc:182]
This TensorFlow binary is optimized to use available CPU instructions in
performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild
TensorFlow with the appropriate compiler flags.
2023-12-21 18:21:56.874750: W
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not
find TensorRT
```

Majd letöltjük és kicsomagoljuk az adathalmazt.

[2]:
```
# A Google Drive-os fájl letöltése
file_id = '1FfgdLCOpDlJZLp0MGohTh1sf4LgHICrN'
url = f'https://drive.google.com/uc?id={file_id}'
output = 'DL_beadandó.zip'
gdown.download(url, output, quiet=False)

# Fájl kicsomagolása
output_folder = 'XRAY'
with zipfile.ZipFile(output, 'r') as zip_ref:
    zip_ref.extractall(output_folder)

# Eredeti ZIP-fájl törlése
os.remove(output)
```

```
Downloading…
From (uriginal):
https://drive.google.com/uc?id=1FfgdLCOpDlJZLp0MGohTh1sf4LgHICrN
From (redirected): https://drive.google.com/uc?id=1FfgdLCOpDlJZLp0MGohTh1sf4LgHI
CrN&confirm=t&uuid=9cd8a302-ddb8-422c-bd2d-0c83c6ab0514
To: /home/kmark7/teszt/DL_beadandó.zip
100%|
              | 412M/412M [00:20<00:00, 20.1MB/s]
```

Képek méretét és batch méretet rakjuk bele egy-egy "fő" változóba.

[3]:
```
IMAGE_SHAPE = (250, 250)
BATCH_SIZE = 10
```

Kialakítjük az adathalmaz struktúráját.

[4]:
```
# Adatok elérési útvonalai
train_dir = "XRAY/train/"
test_dir = "XRAY/test/"

# Kép adatgenerátorok létrehozása
train_datagen = ImageDataGenerator(rescale=1 / 255.)
test_datagen = ImageDataGenerator(rescale=1 / 255.)
```

```
# Tanító adathalmaz betöltése
print("Training images:")
train_data = train_datagen.flow_from_directory(train_dir,
                                               target_size=IMAGE_SHAPE,
                                               batch_size=BATCH_SIZE,
                                               class_mode="categorical")

# Teszt adathalmaz betöltése
print("Testing images:")
test_data = test_datagen.flow_from_directory(test_dir,
                                             target_size=IMAGE_SHAPE,
                                             batch_size=BATCH_SIZE,
                                             class_mode="categorical")
```

```
Training images:
Found 1000 images belonging to 2 classes.
Testing images:
Found 624 images belonging to 2 classes.
```

1.1.2 Maga a CNN architektúráját látjuk itt.

Használunk BatchNormalizálást és Dropout-okat is: először konvolúciós rétegeket használunk, majd ezeket lapítjuk ki.

A konvolúciós rétegek kernelét kcsire (2*2-re) állítottam, a bemeneti képek 250*250 px felbontásúak.

[5]:
```
# Konvolúciós neurális hálózat létrehozása
model = tf.keras.models.Sequential([
    layers.Conv2D(256, (2, 2), activation='relu', input_shape=(250, 250, 3)),
    layers.BatchNormalization(momentum=0.9),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (2, 2), activation='relu'),
    layers.BatchNormalization(momentum=0.9),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(16, (2, 2), activation='relu'),
    layers.BatchNormalization(momentum=0.9),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dropout(0.25),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.9),
    layers.Dense(2, activation='softmax')
])

# Modell adatainak kiíratása
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 249, 249, 256)     3328      
                                                                 
 batch_normalization (Batch  (None, 249, 249, 256)     1024      
 Normalization)                                                  
                                                                 
 max_pooling2d (MaxPooling2  (None, 124, 124, 256)     0         
 D)                                                              
                                                                 
 conv2d_1 (Conv2D)           (None, 123, 123, 64)      65600     
                                                                 
 batch_normalization_1 (Bat  (None, 123, 123, 64)      256       
 chNormalization)                                                
                                                                 
 max_pooling2d_1 (MaxPoolin  (None, 61, 61, 64)        0         
 g2D)                                                            
                                                                 
 conv2d_2 (Conv2D)           (None, 60, 60, 16)        4112      
                                                                 
 batch_normalization_2 (Bat  (None, 60, 60, 16)        64        
 chNormalization)                                                
                                                                 
 max_pooling2d_2 (MaxPoolin  (None, 30, 30, 16)        0         
 g2D)                                                            
                                                                 
 flatten (Flatten)           (None, 14400)             0         
                                                                 
 dropout (Dropout)           (None, 14400)             0         
                                                                 
 dense (Dense)               (None, 64)                921664    
                                                                 
 dropout_1 (Dropout)         (None, 64)                0         
                                                                 
 dense_1 (Dense)             (None, 2)                 130       
                                                                 
=================================================================
Total params: 996178 (3.80 MB)
Trainable params: 995506 (3.80 MB)
Non-trainable params: 672 (2.62 KB)
_________________________________________________________________
2023-12-21 18:24:49.935517: I
external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:901] successful
NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero. See more at
```

```
https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-
pci#L344-L355
2023-12-21 18:24:49.965236: I
external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:901] successful
NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero. See more at
https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-
pci#L344-L355
2023-12-21 18:24:49.965427: I
external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:901] successful
NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero. See more at
https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-
pci#L344-L355
2023-12-21 18:24:49.966929: I
external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:901] successful
NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero. See more at
https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-
pci#L344-L355
2023-12-21 18:24:49.967185: I
external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:901] successful
NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero. See more at
https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-
pci#L344-L355
2023-12-21 18:24:49.967339: I
external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:901] successful
NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero. See more at
https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-
pci#L344-L355
2023-12-21 18:24:50.035936: I
external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:901] successful
NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero. See more at
https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-
pci#L344-L355
2023-12-21 18:24:50.036181: I
external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:901] successful
NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero. See more at
https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-
pci#L344-L355
2023-12-21 18:24:50.036341: I
external/local_xla/xla/stream_executor/cuda/cuda_executor.cc:901] successful
NUMA node read from SysFS had negative value (-1), but there must be at least
one NUMA node, so returning NUMA node zero. See more at
```

```
https://github.com/torvalds/linux/blob/v6.0/Documentation/ABI/testing/sysfs-bus-
pci#L344-L355
2023-12-21 18:24:50.036458: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1929] Created device
/job:localhost/replica:0/task:0/device:GPU:0 with 3435 MB memory:  -> device: 0,
name: Quadro P600, pci bus id: 0000:01:00.0, compute capability: 6.1
```

Excel fájlba logolunk, mivel később ott rekonstruálni tudjuk a tanulási görbéket, grafikonokat, és ezeken kívül egyéb statisztikákat is készíthetünk.

[6]:
```
csv_logger = CSVLogger('ModelLog.csv', separator=',', append=False)
```

Beállítjuk az optimalizálót és a model aktivációs függvényét, tanulási rátáját.

[7]:
```
initial_learning_rate = 0.00001
lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate, decay_steps=5000, decay_rate=0.9, staircase=True)

model.compile(loss='categorical_crossentropy',
              optimizer=tf.keras.optimizers.AdamW(learning_rate=lr_schedule),
              metrics=['accuracy'])
```

1.1.3 Bővítjük az adathalmazt.

[8]:
```
# Bővített adathalmaz generátor létrehozása
train_datagen_augmented = ImageDataGenerator(
    rescale=1 / 255.,
    rotation_range=40,
    shear_range=0.3,
    zoom_range=0.1,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True
)

# Bővített tanító adathalmaz generálása
augmented_train_data = train_datagen_augmented.flow_from_directory(
    train_dir,
    target_size=IMAGE_SHAPE,
    batch_size=BATCH_SIZE,
    class_mode="categorical",
    shuffle=True
)

# Tanító adathalmazok összekapcsolása
combined_train_data = tf.data.Dataset.from_generator(
    lambda: train_data,
    output_signature=(
        tf.TensorSpec(shape=(None, 250, 250, 3), dtype=tf.float32),
        tf.TensorSpec(shape=(None, 2), dtype=tf.float32)
    )
).concatenate(
    tf.data.Dataset.from_generator(
        lambda: augmented_train_data,
        output_signature=(
            tf.TensorSpec(shape=(None, 250, 250, 3), dtype=tf.float32),
            tf.TensorSpec(shape=(None, 2), dtype=tf.float32)
        )
    )
)
```

```
Found 1000 images belonging to 2 classes.
```

Elmentünk egy tanítatlan modelt, erre majd a későbbiekben lesz szükségünk.

[9]:
```
model.save("PheoModel.h5")
```

```
/home/kmark7/pysorflow/lib/python3.11/site-
packages/keras/src/engine/training.py:3103: UserWarning: You are saving your
model as an HDF5 file via `model.save()`. This file format is considered legacy.
We recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
  saving_api.save_model(
```

1.1.4 Korai leállítást és modell mentés callback-eket inicializáljuk.

Ez early stopping helyett ModelCheckpoint-ot használunk: minden epoch után ideiglenesen elmentjük, és a végén az adott szempont szerint legjobb modelt mentjük el véglegesen.

[10]:
```
# Korai leállítás
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=5,
                               restore_best_weights=True,
                               verbose=1)

# Modell mentés callback-ek inicializálása
model_checkpoint = ModelCheckpoint("PheoModel.h5",
                                   save_best_only=True,
                                   monitor='val_accuracy',
                                   mode='max',
                                   verbose=1)
```

1.1.5 Tanítjuk a modellt!

```
[11]: loaded_model = load_model("PheoModel.h5")
history = loaded_model.fit(combined_train_data,
                           epochs=50,
                           steps_per_epoch=len(train_data),
                           validation_data=test_data,
                           validation_steps=len(test_data),
                           callbacks=[early_stopping, model_checkpoint,␣
 ↪csv_logger])
```

```
Epoch 1/50
```

```
2023-12-21 18:25:17.347378: I
external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:454] Loaded cuDNN
version 8907
2023-12-21 18:25:19.859691: W
external/local_tsl/tsl/framework/bfc_allocator.cc:296] Allocator (GPU_0_bfc) ran
out of memory trying to allocate 2.36GiB with freed_by_count=0. The caller
indicates that this is not a failure, but this may mean that there could be
performance gains if more memory were available.
2023-12-21 18:25:20.399352: W
external/local_tsl/tsl/framework/bfc_allocator.cc:296] Allocator (GPU_0_bfc) ran
out of memory trying to allocate 2.36GiB with freed_by_count=0. The caller
indicates that this is not a failure, but this may mean that there could be
performance gains if more memory were available.
2023-12-21 18:25:21.127356: I external/local_xla/xla/service/service.cc:168] XLA
service 0x7f831d916060 initialized for platform CUDA (this does not guarantee
that XLA will be used). Devices:
2023-12-21 18:25:21.127380: I external/local_xla/xla/service/service.cc:176]
StreamExecutor device (0): Quadro P600, Compute Capability 6.1
2023-12-21 18:25:21.132673: I
tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:269] disabling MLIR
crash reproducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTORY` to enable.
WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
I0000 00:00:1703179521.199814  126554 device_compiler.h:186] Compiled cluster
using XLA!  This line is logged at most once for the lifetime of the process.
2023-12-21 18:25:23.180889: W
external/local_tsl/tsl/framework/bfc_allocator.cc:296] Allocator (GPU_0_bfc) ran
out of memory trying to allocate 2.08GiB with freed_by_count=0. The caller
indicates that this is not a failure, but this may mean that there could be
performance gains if more memory were available.
2023-12-21 18:25:23.715751: W
external/local_tsl/tsl/framework/bfc_allocator.cc:296] Allocator (GPU_0_bfc) ran
out of memory trying to allocate 2.08GiB with freed_by_count=0. The caller
indicates that this is not a failure, but this may mean that there could be
performance gains if more memory were available.
```

```
100/100 [==============================] - ETA: 0s - loss: 2.5697 - accuracy:
0.5300
Epoch 1: val_accuracy improved from -inf to 0.72115, saving model to
PheoModel.h5
100/100 [==============================] - 47s 384ms/step - loss: 2.5697 -
accuracy: 0.5300 - val_loss: 0.5571 - val_accuracy: 0.7212
Epoch 2/50
100/100 [==============================] - ETA: 0s - loss: 1.5962 - accuracy:
0.5840
Epoch 2: val_accuracy improved from 0.72115 to 0.80128, saving model to
PheoModel.h5
100/100 [==============================] - 36s 364ms/step - loss: 1.5962 -
accuracy: 0.5840 - val_loss: 0.4556 - val_accuracy: 0.8013
Epoch 3/50
100/100 [==============================] - ETA: 0s - loss: 0.9382 - accuracy:
0.6630
Epoch 3: val_accuracy improved from 0.80128 to 0.84455, saving model to
PheoModel.h5
100/100 [==============================] - 37s 366ms/step - loss: 0.9382 -
accuracy: 0.6630 - val_loss: 0.4217 - val_accuracy: 0.8446
Epoch 4/50
100/100 [==============================] - ETA: 0s - loss: 0.7260 - accuracy:
0.6860
Epoch 4: val_accuracy improved from 0.84455 to 0.85096, saving model to
PheoModel.h5
100/100 [==============================] - 36s 364ms/step - loss: 0.7260 -
accuracy: 0.6860 - val_loss: 0.4190 - val_accuracy: 0.8510
Epoch 5/50
100/100 [==============================] - ETA: 0s - loss: 0.6820 - accuracy:
0.7060
Epoch 5: val_accuracy improved from 0.85096 to 0.86218, saving model to
PheoModel.h5
100/100 [==============================] - 36s 365ms/step - loss: 0.6820 -
accuracy: 0.7060 - val_loss: 0.4179 - val_accuracy: 0.8622
Epoch 6/50
100/100 [==============================] - ETA: 0s - loss: 0.5993 - accuracy:
0.7040
Epoch 6: val_accuracy improved from 0.86218 to 0.86538, saving model to
PheoModel.h5
100/100 [==============================] - 36s 364ms/step - loss: 0.5993 -
accuracy: 0.7040 - val_loss: 0.4110 - val_accuracy: 0.8654
Epoch 7/50
100/100 [==============================] - ETA: 0s - loss: 0.5417 - accuracy:
0.7380
Epoch 7: val_accuracy improved from 0.86538 to 0.86859, saving model to
PheoModel.h5
100/100 [==============================] - 36s 364ms/step - loss: 0.5417 -
accuracy: 0.7380 - val_loss: 0.3904 - val_accuracy: 0.8686
```

```
Epoch 8/50
100/100 [==============================] - ETA: 0s - loss: 0.5300 - accuracy:
0.7310
Epoch 8: val_accuracy improved from 0.86859 to 0.88782, saving model to
PheoModel.h5
100/100 [==============================] - 36s 365ms/step - loss: 0.5300 -
accuracy: 0.7310 - val_loss: 0.3689 - val_accuracy: 0.8878
Epoch 9/50
100/100 [==============================] - ETA: 0s - loss: 0.4784 - accuracy:
0.7630
Epoch 9: val_accuracy improved from 0.88782 to 0.89103, saving model to
PheoModel.h5
100/100 [==============================] - 36s 364ms/step - loss: 0.4784 -
accuracy: 0.7630 - val_loss: 0.3516 - val_accuracy: 0.8910
Epoch 10/50
100/100 [==============================] - ETA: 0s - loss: 0.4508 - accuracy:
0.7830
Epoch 10: val_accuracy improved from 0.89103 to 0.89263, saving model to
PheoModel.h5
100/100 [==============================] - 36s 364ms/step - loss: 0.4508 -
accuracy: 0.7830 - val_loss: 0.3391 - val_accuracy: 0.8926
Epoch 11/50
100/100 [==============================] - ETA: 0s - loss: 0.4518 - accuracy:
0.7890
Epoch 11: val_accuracy improved from 0.89263 to 0.89904, saving model to
PheoModel.h5
100/100 [==============================] - 36s 365ms/step - loss: 0.4518 -
accuracy: 0.7890 - val_loss: 0.3257 - val_accuracy: 0.8990
Epoch 12/50
100/100 [==============================] - ETA: 0s - loss: 0.4322 - accuracy:
0.7950
Epoch 12: val_accuracy did not improve from 0.89904
100/100 [==============================] - 36s 364ms/step - loss: 0.4322 -
accuracy: 0.7950 - val_loss: 0.3190 - val_accuracy: 0.8910
Epoch 13/50
100/100 [==============================] - ETA: 0s - loss: 0.4381 - accuracy:
0.8020
Epoch 13: val_accuracy did not improve from 0.89904
100/100 [==============================] - 37s 367ms/step - loss: 0.4381 -
accuracy: 0.8020 - val_loss: 0.3061 - val_accuracy: 0.8926
Epoch 14/50
100/100 [==============================] - ETA: 0s - loss: 0.4131 - accuracy:
0.8160
Epoch 14: val_accuracy did not improve from 0.89904
100/100 [==============================] - 37s 366ms/step - loss: 0.4131 -
accuracy: 0.8160 - val_loss: 0.2918 - val_accuracy: 0.8990
Epoch 15/50
100/100 [==============================] - ETA: 0s - loss: 0.3813 - accuracy:
```

```
0.8210
Epoch 15: val_accuracy did not improve from 0.89904
100/100 [==============================] - 36s 365ms/step - loss: 0.3813 -
accuracy: 0.8210 - val_loss: 0.2937 - val_accuracy: 0.8910
Epoch 16/50
100/100 [==============================] - ETA: 0s - loss: 0.3641 - accuracy:
0.8180Restoring model weights from the end of the best epoch: 11.

Epoch 16: val_accuracy did not improve from 0.89904
100/100 [==============================] - 36s 364ms/step - loss: 0.3641 -
accuracy: 0.8180 - val_loss: 0.3011 - val_accuracy: 0.8846
Epoch 16: early stopping
```

1.1.6 Ábrázoljuk a pontosság -és a veszteségfüggvényeket.

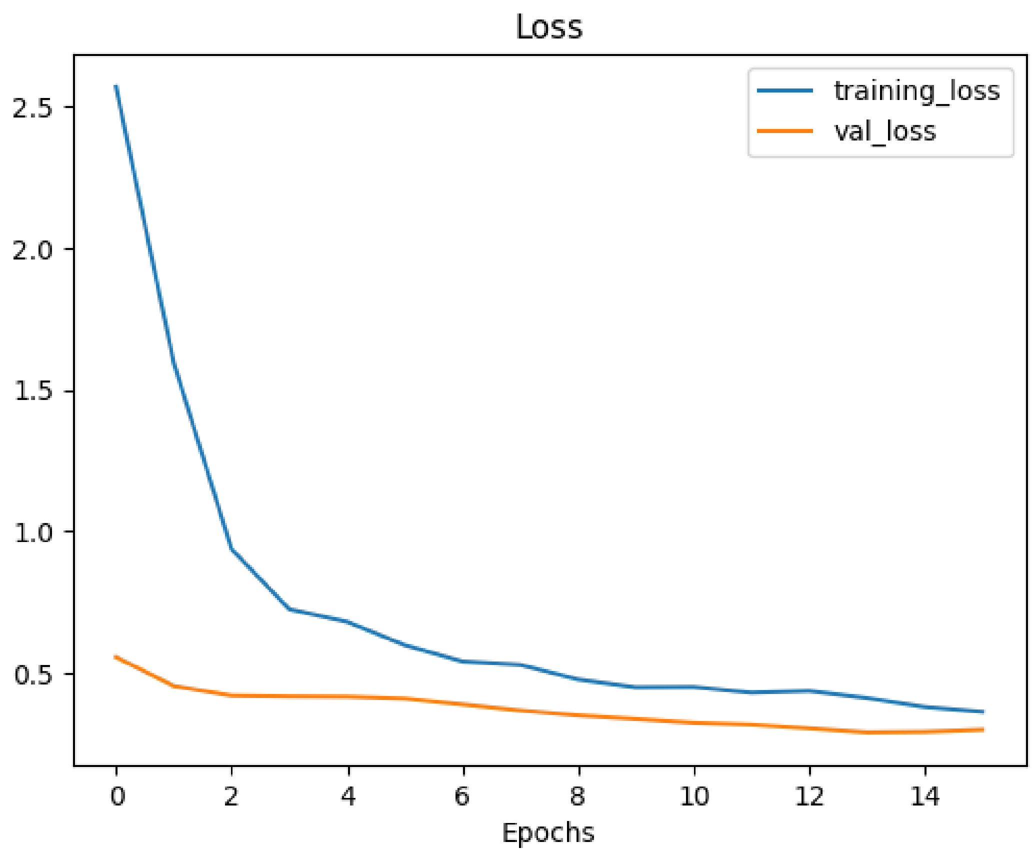
A modellünk viszonylag stabilan és egyenletesen tanul, de elég lassan is sajnos.

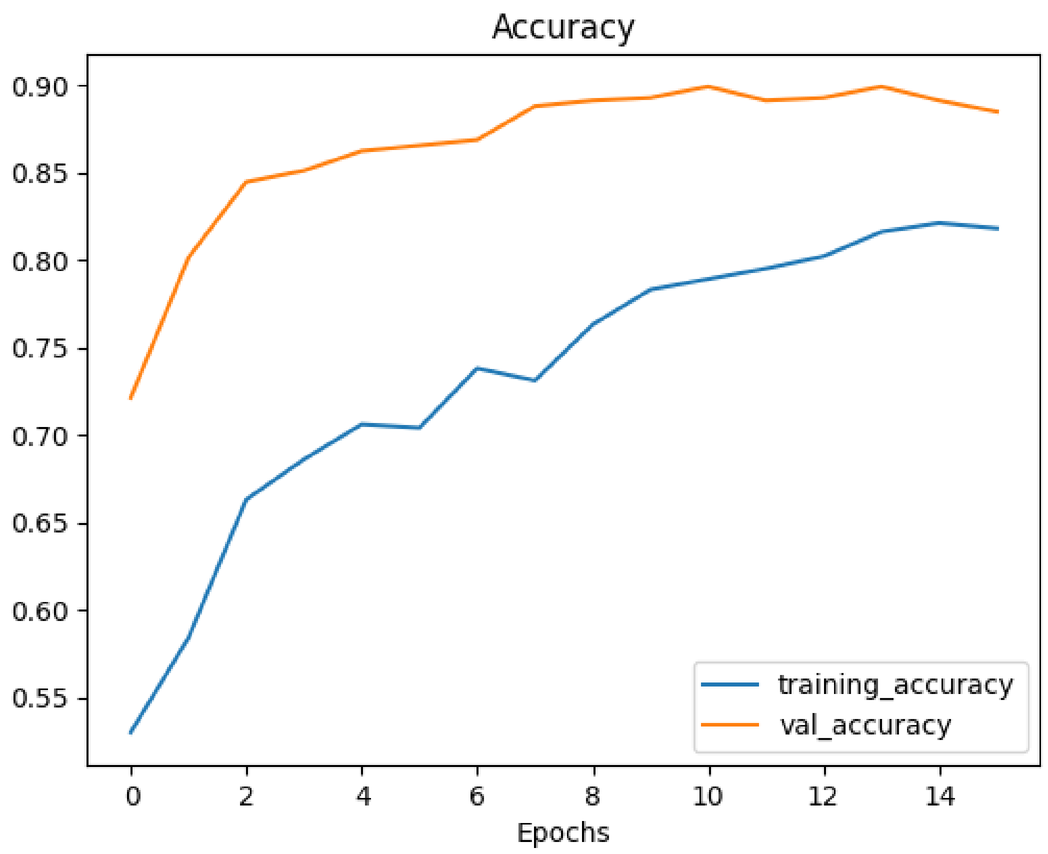
[12]:
```python
def plot_loss_curves(history):
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    accuracy = history.history['accuracy']
    val_accuracy = history.history['val_accuracy']
    epochs = range(len(history.history['loss']))

    plt.plot(epochs, loss, label='training_loss')
    plt.plot(epochs, val_loss, label='val_loss')
    plt.title('Loss')
    plt.xlabel('Epochs')
    plt.legend()
    plt.show()

    plt.figure()
    plt.plot(epochs, accuracy, label='training_accuracy')
    plt.plot(epochs, val_accuracy, label='val_accuracy')
    plt.title('Accuracy')
    plt.xlabel('Epochs')
    plt.legend()
    plt.show()


plot_loss_curves(history)
```


Loss
training_loss
val_loss
2.5
2.0
1.5
1.0
0.5
0
2
4
6
8
10
12
14
Epochs



Kiíratjuk a legjobb modellnek a pontosságát, amit elmentettünk.

```
[18]: model2 = tf.keras.models.load_model("PheoModel.h5")
      print("A modell teszthalmazon vett pontossága:", round(max(history.
       ↪history['val_accuracy'])*100, 5), "%")
```

```
A modell teszthalmazon vett pontossága: 89.90384 %
```

```
[ ]:
```