Návod použití

Google Colab

Pokud se nacházíte v rozhraní Google Colab, složka "DMP_Neuronove_site", ve které se nachází tento notebook a program "custom_ann.py", by se měla na Vašem Google Disku nacházet ve výchozí složce "Můj Disk".

všechny buňky lze naráz spustit pomocí "Běh -> Spustit vše" v horní liště. Samostatné buňky lze spustit pomocí kliknutím na danou buňku a klávesami "Ctrl+Enter".

Chvilku možná bude trvat, než se notebook připojí k běhovému prostředí, které doporučuji nastavit na vzdálenou GPU pomocí "Běh -> Zvolit běhové prostředí -> GPU".

Program v určité části požádá o přístup k Vašemu Google Disku, aby mohl naimportovat neuronovou síť ze souboru "custom_cnn.py". Pokud povolíte, zbytek programu by měl proběhnout bez problému. Pokud se po udělení přístupu notebook zasekne, odpojte a smažte běh pomocí "Běh -> Odpojit a smazat běh" a spusťte manuálně buňku po buňce.

Jiné rozhraní

Pokud se nacházíte v jiném rozhraní, pak zakomentujte nebo smažte všechny řádky kódu okomentované "# Pro Colab" a ujistěte se, že všechny použité knihovny máte nainstalované na počítači nebo ve Vašem virtuálním prostředí.

Pro správnou funkci musíte být připojeni k internetu, protože datový soubor GTSRB je nahráván z cloudové databáze knihovny Deeplake.

Instalace modulu Deeplake do běhového prostředí

```
!pip install deeplake # Pro Colab
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting deeplake
  Downloading deeplake-3.2.18.tar.gz (447 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 447.8/447.8 KB 4.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (from deeplake) (1.22.4)
Requirement already satisfied: pillow in /usr/local/lib/python3.9/dist-packages (from deeplake) (8.4.0)
Collecting boto3
  Downloading boto3-1.26.99-py3-none-any.whl (135 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 135.5/135.5 KB 12.9 MB/s eta 0:00:00
Requirement already satisfied: click in /usr/local/lib/python3.9/dist-packages (from deeplake) (8.1.3)
Collecting pathos
  Downloading pathos-0.3.0-py3-none-any.whl (79 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.8/79.8 KB 10.2 MB/s eta 0:00:00
Collecting humbug>=0.2.6
  Downloading humbug-0.3.0-py3-none-any.whl (14 kB)
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages (from deeplake) (4.65.0)
Collecting numcodecs
  Downloading numcodecs-0.11.0-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (6.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.7/6.7 MB 68.3 MB/s eta 0:00:00
Collecting pyjwt
  Downloading PyJWT-2.6.0-py3-none-any.whl (20 kB)
Collecting aioboto3==10.4.0
  Downloading aioboto3-10.4.0-py3-none-any.whl (32 kB)
Requirement already satisfied: nest_asyncio in /usr/local/lib/python3.9/dist-packages (from deeplake) (1.5.6)
Collecting aiobotocore[boto3]==2.4.2
  Downloading aiobotocore-2.4.2-py3-none-any.whl (66 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 66.8/66.8 KB 8.1 MB/s eta 0:00:00
Requirement already satisfied: wrapt>=1.10.10 in /usr/local/lib/python3.9/dist-packages (from aiobotocore[boto3]==2.4.2->aioboto3
Collecting aiohttp>=3.3.1
  Downloading aiohttp-3.8.4-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.0/1.0 MB 68.2 MB/s eta 0:00:00
Collecting botocore<1.27.60,>=1.27.59
  Downloading botocore-1.27.59-py3-none-any.whl (9.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.1/9.1 MB 98.2 MB/s eta 0:00:00
Collecting aioitertools>=0.5.1
  Downloading aioitertools-0.11.0-py3-none-any.whl (23 kB)
Collecting boto3
  Downloading boto3-1.24.59-py3-none-any.whl (132 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 132.5/132.5 KB 15.9 MB/s eta 0:00:00
Collecting jmespath<2.0.0,>=0.7.1
  Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)
Collecting s3transfer<0.7.0,>=0.6.0
  Downloading s3transfer-0.6.0-py3-none-any.whl (79 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.6/79.6 KB 11.4 MB/s eta 0:00:00
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages (from humbug>=0.2.6->deeplake) (2.27.1)
Requirement already satisfied: entrypoints in /usr/local/lib/python3.9/dist-packages (from numcodecs->deeplake) (0.4)
Collecting multiprocess>=0.70.14
  Downloading multiprocess-0.70.14-py39-none-any.whl (132 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 132.9/132.9 KB 18.0 MB/s eta 0:00:00
```

```
Collecting ppft>=1.7.6.6
  Downloading ppft-1.7.6.6-py3-none-any.whl (52 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 52.8/52.8 KB 6.1 MB/s eta 0:00:00
Collecting dill>=0.3.6
  Downloading dill-0.3.6-py3-none-any.whl (110 kB)
```

Importování knihoven

```
import deeplake
import os
import cv2
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
```

Importování vlastní neuronové sítě z google drive do google colab

```
from google.colab import drive # Pro Colab
drive.mount('/content/gdrive') # Pro Colab
```

```
Mounted at /content/gdrive
```

```
import sys # Pro Colab
sys.path.append('/content/gdrive/My Drive/DMP_Neuronove_site') # Pro Colab
```

```
from custom_ann import Artificial_Neural_Network
```

Nahrání a zpracování dat

```
test_data = deeplake.load("hub://activeloop/gtsrb-test")
training_data = deeplake.load("hub://activeloop/gtsrb-train")
```

```
hub://activeloop/gtsrb-test loaded successfully.

This dataset can be visualized in Jupyter Notebook by ds.visualize() or at https://app.activeloop.ai/activeloop/gtsrb-test

hub://activeloop/gtsrb-train loaded successfully.

This dataset can be visualized in Jupyter Notebook by ds.visualize() or at https://app.activeloop.ai/activeloop/gtsrb-train
```

```
# Načtení trénovacích a testovacích vstupních dat
X_test = []
X_train = []
for image in test_data.images:
    image = image.numpy()
    image = cv2.resize(image, (30, 30))
    X_test.append(image)
for image in training_data.images:
    image = image.numpy()
    image = cv2.resize(image, (30, 30))
    X_train.append(image)
X_train = np.array(X_train)
X_test = np.array(X_test)

# Načtení trénovacích a testovacích výstupních dat
y_test = np.array(test_data.labels.numpy().reshape(1,len(test_data.labels.numpy()))[0])
y_train = np.array(training_data.labels.numpy().reshape(1,len(training_data.labels.numpy()))[0])

# Získání validačních dat
X_valid, X_test = X_test[:3000], X_test[3000:]
y_valid, y_test = y_test[:3000], y_test[3000:]

# Normalizace vstupních dat do hodnot v rozmezí 0-1
X_train_norm, X_test_norm, X_valid_norm = X_train / 255, X_test / 255, X_valid / 255

# Manuální zploštění snímků pro mou vlastní neuronovou síť
X_train_flattened = [np.array([x.flatten()]).T for x in X_train_norm]
X_test_flattened = [np.array([x.flatten()]).T for x in X_test_norm]
X_valid_flattened = [np.array([x.flatten()]).T for x in X_valid_norm]

# Kódování výstupních dat do cílových vektorů
```

```
y_train_encoded = tf.keras.utils.to_categorical(y_train)
y_test_encoded = tf.keras.utils.to_categorical(y_test)
y_valid_encoded = tf.keras.utils.to_categorical(y_valid)
```

Ilustrační snímky datového souboru GTSRB

```
class_names = training_data.labels.info.class_names
```

```
fig, axes = plt.subplots(5, 7, figsize = (20, 17))
for row in axes:
  for axe in row:
    i = np.random.randint(len(X_train))
    axe.imshow(X_train[i])
    axe.set_title(class_names[y_train[i]])
```

Učení mé vlastní neuronové sítě

```
custom_model = Artificial_Neural_Network((2700, 128, 64, 43))
custom_model.stochastic_gradient_descent(X_train_flattened, y_train_encoded,
                                         epochs=30, mini_batch_size=10, learning_rate=0.5,
                                         validation_data=(X_valid_flattened, y_valid_encoded))
```

```
Epoch 1 -> Training loss: 2.7405106862758775, Training accuracy: 0.017444974368129765, Validation loss: 2.7441513346975164, Validat
Epoch 2 -> Training loss: 2.622351852174909, Training accuracy: 0.018873217883649162, Validation loss: 2.656647999355275, Validatio
Epoch 3 -> Training loss: 0.6866642416626021, Training accuracy: 0.4467851768726568, Validation loss: 0.7273554726852325, Validatio
```

```
Epoch 4 -> Training loss: 0.6012312316647056, Training accuracy: 0.5221505266647963, Validation loss: 0.6600324791650656, Validatio
Epoch 5 -> Training loss: 0.5809022219057733, Training accuracy: 0.5475528577622485, Validation loss: 0.6366886915224054, Validatio
Epoch 6 -> Training loss: 0.5057240281676789, Training accuracy: 0.6230202249483537, Validation loss: 0.5755552513410986, Validatio
Epoch 7 -> Training loss: 0.4499861737773911, Training accuracy: 0.669616669642174, Validation loss: 0.5295734333294619, Validation
Epoch 8 -> Training loss: 0.42297604936447947, Training accuracy: 0.6903006962687138, Validation loss: 0.5019020990294673, Validati
Epoch 9 -> Training loss: 0.3628217272754018, Training accuracy: 0.7354688974470147, Validation loss: 0.4660257227989563, Validatio
Epoch 10 -> Training loss: 0.3765502465887564, Training accuracy: 0.728710245096789, Validation loss: 0.4414410929812737, Validatio
Epoch 11 -> Training loss: 0.3403165453618467, Training accuracy: 0.7554132979673034, Validation loss: 0.4460553137247248, Validati
Epoch 12 -> Training loss: 0.39585312745955564, Training accuracy: 0.7139687316687495, Validation loss: 0.48553185419917244, Valida
Epoch 13 -> Training loss: 0.3193500772592372, Training accuracy: 0.7723481853656048, Validation loss: 0.4393987632030334, Validati
Epoch 14 -> Training loss: 0.35160657975123744, Training accuracy: 0.7450585324797878, Validation loss: 0.4399518420131911, Validat
Epoch 15 -> Training loss: 0.3046286235393671, Training accuracy: 0.7780866637761739, Validation loss: 0.3963323880555933, Validati
Epoch 16 -> Training loss: 0.28397243745288786, Training accuracy: 0.7930067076436532, Validation loss: 0.3853176870485674, Validat
Epoch 17 -> Training loss: 0.30286876180791894, Training accuracy: 0.7799994899130301, Validation loss: 0.39769240693177327, Valida
Epoch 18 -> Training loss: 0.3151100400784709, Training accuracy: 0.7741589941084955, Validation loss: 0.4223475001147877, Validati
Epoch 19 -> Training loss: 0.2844437235792998, Training accuracy: 0.7963222729475375, Validation loss: 0.3972709904845311, Validati
Epoch 20 -> Training loss: 0.2777502345091203, Training accuracy: 0.7990767425846107, Validation loss: 0.41590684103868625, Validat
Epoch 21 -> Training loss: 0.27806544558843804, Training accuracy: 0.8002244382667245, Validation loss: 0.39685029584802667, Valida
Epoch 22 -> Training loss: 0.2694553845256716, Training accuracy: 0.8032849600856946, Validation loss: 0.40152020485278217, Validat
Epoch 23 -> Training loss: 0.2841327487852109, Training accuracy: 0.7937718380983958, Validation loss: 0.39279036366885794, Validat
Epoch 24 -> Training loss: 0.27514821322647404, Training accuracy: 0.8002244382667245, Validation loss: 0.3833874655114797, Validat
Epoch 25 -> Training loss: 0.2550866660588046, Training accuracy: 0.8130276212094162, Validation loss: 0.3561826931790581, Validati
Epoch 26 -> Training loss: 0.25773488309104947, Training accuracy: 0.8089214211022979, Validation loss: 0.3726706842124441, Validat
Epoch 27 -> Training loss: 0.254983145152814, Training accuracy: 0.8134866994822617, Validation loss: 0.3809388586170378, Validatio
Epoch 28 -> Training loss: 0.24345659120665664, Training accuracy: 0.8204238822719274, Validation loss: 0.3681511597470344, Validat
Epoch 29 -> Training loss: 0.23876009111722912, Training accuracy: 0.8226937692876636, Validation loss: 0.3716048918410283, Validat
Epoch 30 -> Training loss: 0.2306893968068797, Training accuracy: 0.8337116478359561, Validation loss: 0.3483132874258122, Validati
```

Vytvoření a učení neuronové sítě pomocí TensorFlow Keras

```
keras_model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(64, (3, 3), activation="relu",
                           input_shape=(30, 30, 3)),
    tf.keras.layers.Conv2D(64, (3, 3), activation="relu"),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Conv2D(128, (3, 3), activation="relu"),
    tf.keras.layers.Conv2D(128, (3, 3), activation="relu"),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Conv2D(256, (3, 3), activation="relu"),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation="relu"),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(43, activation="softmax")
])
```

```
keras_model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_15 (Conv2D)          (None, 28, 28, 64)        1792

 conv2d_16 (Conv2D)          (None, 26, 26, 64)        36928

 max_pooling2d_9 (MaxPooling  (None, 13, 13, 64)       0
 2D)

 dropout_12 (Dropout)        (None, 13, 13, 64)        0

 conv2d_17 (Conv2D)          (None, 11, 11, 128)       73856

 conv2d_18 (Conv2D)          (None, 9, 9, 128)         147584

 max_pooling2d_10 (MaxPoolin  (None, 4, 4, 128)        0
 g2D)

 dropout_13 (Dropout)        (None, 4, 4, 128)         0

 conv2d_19 (Conv2D)          (None, 2, 2, 256)         295168

 max_pooling2d_11 (MaxPoolin  (None, 1, 1, 256)        0
 g2D)

 dropout_14 (Dropout)        (None, 1, 1, 256)         0

 flatten_3 (Flatten)         (None, 256)               0

 dense_6 (Dense)             (None, 128)               32896
```

```
 dropout_15 (Dropout)        (None, 128)               0

 dense_7 (Dense)             (None, 43)                5547

=================================================================
Total params: 593,771
Trainable params: 593,771
Non-trainable params: 0
_________________________________________________________________
```

```
keras_model.compile(optimizer = "nadam",
                    loss = "categorical_crossentropy",
                    metrics=["accuracy"])
```

```
keras_model_history = keras_model.fit(X_train_norm, y_train_encoded, epochs=30, validation_data=(X_valid_norm, y_valid_encoded))
```

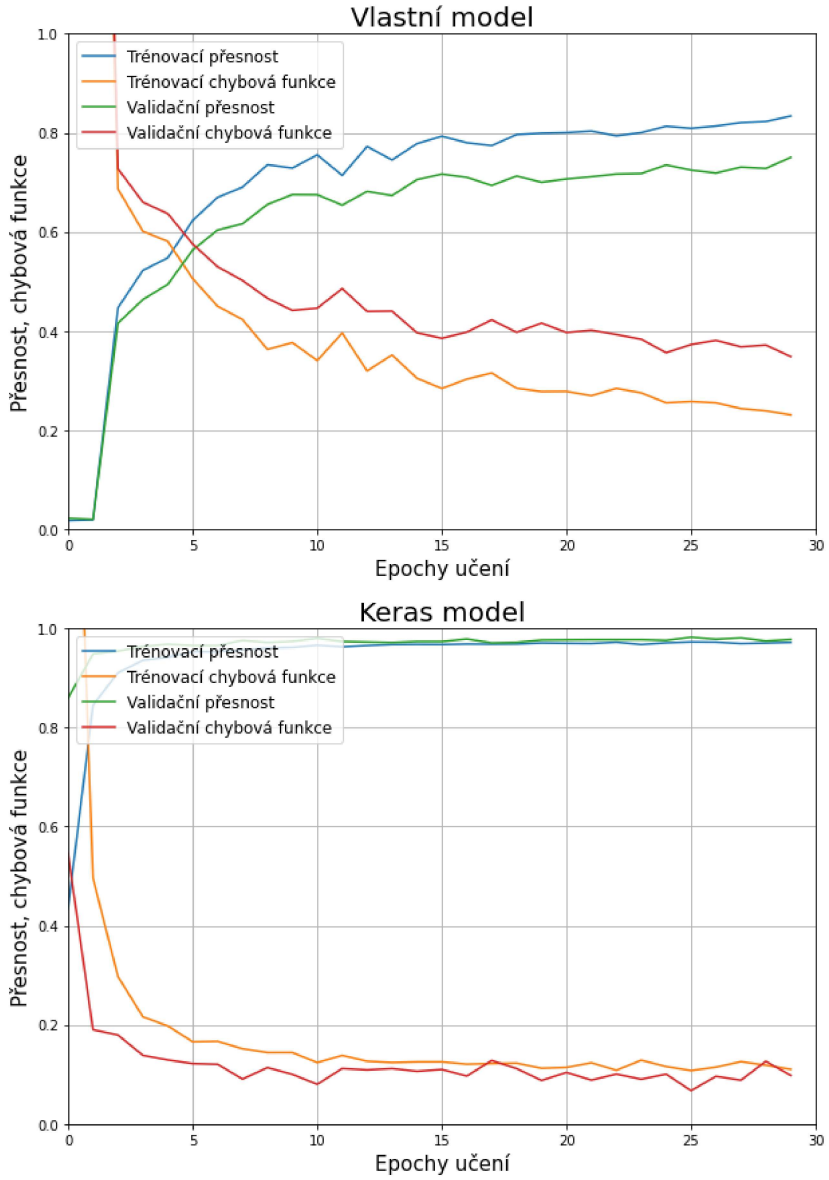
```
Epoch 1/30
1226/1226 [==============================] - 16s 9ms/step - loss: 1.9537 - accuracy: 0.4344 - val_loss: 0.5461 - val_accuracy: 0.
Epoch 2/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.4961 - accuracy: 0.8449 - val_loss: 0.1900 - val_accuracy: 0.
Epoch 3/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.2975 - accuracy: 0.9104 - val_loss: 0.1794 - val_accuracy: 0.
Epoch 4/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.2163 - accuracy: 0.9353 - val_loss: 0.1384 - val_accuracy: 0.
Epoch 5/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1976 - accuracy: 0.9416 - val_loss: 0.1291 - val_accuracy: 0.
Epoch 6/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1657 - accuracy: 0.9529 - val_loss: 0.1217 - val_accuracy: 0.
Epoch 7/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1665 - accuracy: 0.9529 - val_loss: 0.1204 - val_accuracy: 0.
Epoch 8/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1516 - accuracy: 0.9563 - val_loss: 0.0904 - val_accuracy: 0.
Epoch 9/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1440 - accuracy: 0.9603 - val_loss: 0.1136 - val_accuracy: 0.
Epoch 10/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1440 - accuracy: 0.9613 - val_loss: 0.0999 - val_accuracy: 0.
Epoch 11/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1238 - accuracy: 0.9658 - val_loss: 0.0798 - val_accuracy: 0.
Epoch 12/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1382 - accuracy: 0.9626 - val_loss: 0.1118 - val_accuracy: 0.
Epoch 13/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1265 - accuracy: 0.9651 - val_loss: 0.1089 - val_accuracy: 0.
Epoch 14/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1239 - accuracy: 0.9672 - val_loss: 0.1117 - val_accuracy: 0.
Epoch 15/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1252 - accuracy: 0.9676 - val_loss: 0.1061 - val_accuracy: 0.
Epoch 16/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1252 - accuracy: 0.9673 - val_loss: 0.1098 - val_accuracy: 0.
Epoch 17/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1206 - accuracy: 0.9683 - val_loss: 0.0965 - val_accuracy: 0.
Epoch 18/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1219 - accuracy: 0.9680 - val_loss: 0.1280 - val_accuracy: 0.
Epoch 19/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1226 - accuracy: 0.9683 - val_loss: 0.1116 - val_accuracy: 0.
Epoch 20/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1122 - accuracy: 0.9705 - val_loss: 0.0876 - val_accuracy: 0.
Epoch 21/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1139 - accuracy: 0.9699 - val_loss: 0.1038 - val_accuracy: 0.
Epoch 22/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1233 - accuracy: 0.9694 - val_loss: 0.0884 - val_accuracy: 0.
Epoch 23/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1082 - accuracy: 0.9721 - val_loss: 0.1004 - val_accuracy: 0.
Epoch 24/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1285 - accuracy: 0.9674 - val_loss: 0.0901 - val_accuracy: 0.
Epoch 25/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1157 - accuracy: 0.9707 - val_loss: 0.1003 - val_accuracy: 0.
Epoch 26/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1076 - accuracy: 0.9726 - val_loss: 0.0670 - val_accuracy: 0.
Epoch 27/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1145 - accuracy: 0.9722 - val_loss: 0.0960 - val_accuracy: 0.
Epoch 28/30
1226/1226 [==============================] - 11s 9ms/step - loss: 0.1257 - accuracy: 0.9693 - val_loss: 0.0881 - val_accuracy: 0.
Epoch 29/30
```

Testování a porovnání výsledných modelů

Porovnání průběhu učení neuronových sítí

```
```

```
histories = {"Vlastní model": custom_model.history, "Keras model": keras_model_history.history}
fig, axes = plt.subplots(len(histories), 1, figsize = (10, 15))
for index, (title, history) in enumerate(histories.items()):
  axes[index].set_title(title, fontsize=20)
  axes[index].set_xlabel('Epochy učení', fontsize=15)
  axes[index].set_ylabel('Přesnost, chybová funkce', fontsize=15)
  axes[index].plot(history["accuracy"])
  axes[index].plot(history["loss"])
  axes[index].plot(history["val_accuracy"])
  axes[index].plot(history["val_loss"])
  axes[index].grid(True)
  axes[index].set_xlim(0, len(history["accuracy"]))
  axes[index].set_ylim(0, 1)
  axes[index].legend(["Trénovací přesnost", "Trénovací chybová funkce", "Validační přesnost",  "Validační chybová funkce"], loc="upper le
```



Testování modelů

```
custom_model.evaluate(X_test_flattened, y_test_encoded)
```

```
Test loss: 0.33786788928962036, Test accuracy: 0.7583199449162268
```

```
keras_model.evaluate(X_test_norm, y_test_encoded)
```

```
137/137 [==============================] - 1s 5ms/step - loss: 0.0767 - accuracy: 0.9819
[0.07674568891525269, 0.9818682670593262]
```

Ilustrační predikce

Ilustrační predikce mé vlastní neuronové sítě

```
predictions = custom_model.predict(X_test_flattened).argmax(axis=1)
predictions
```

```
array([13,  5, 30, ..., 11,  2,  2])
```

Správné predikce

```
fig, axes = plt.subplots(3, 4, figsize = (30, 25))
for row in axes:
  for axe in row:
    i = np.random.randint(len(predictions))
    while y_test[i] != predictions[i]:
      i = np.random.randint(len(predictions))
    axe.imshow(X_test[i])
    axe.set_title(f"Správné označení: {class_names[y_test[i]]}\n Predikce: {class_names[predictions[i]]}", fontsize=20)
```

Špatné predikce

```
fig, axes = plt.subplots(3, 4, figsize = (30, 25))
for row in axes:
  for axe in row:
    i = np.random.randint(len(predictions))
    while y_test[i] == predictions[i]:
      i = np.random.randint(len(predictions))
    axe.imshow(X_test[i])
    axe.set_title(f"Správné označení: {class_names[y_test[i]]}\n Predikce: {class_names[predictions[i]]}", fontsize=20)
```

Ilustrační predikce neuronové sítě vytvořené pomocí TensorFlow Keras

```
predictions = keras_model.predict(X_test).argmax(axis=1)
predictions
```

```
137/137 [==============================] - 0s 2ms/step
array([13,  5, 30, ..., 11,  2, 34])
```

Správné predikce

```
fig, axes = plt.subplots(3, 4, figsize = (30, 25))
for row in axes:
  for axe in row:
    i = np.random.randint(len(predictions))
    while y_test[i] != predictions[i]:
      i = np.random.randint(len(predictions))
    axe.imshow(X_test[i])
    axe.set_title(f"Správné označení: {class_names[y_test[i]]}\n Predikce: {class_names[predictions[i]]}", fontsize=20)
```

Špatné predikce

```
fig, axes = plt.subplots(3, 4, figsize = (30, 25))
for row in axes:
  for axe in row:
    i = np.random.randint(len(predictions))
```

```
    while y_test[i] == predictions[i]:
      i = np.random.randint(len(predictions))
    axe.imshow(X_test[i])
    axe.set_title(f"Správné označení: {class_names[y_test[i]]}\n Predikce: {class_names[predictions[i]]}", fontsize=20)
```

Exportování konečného modelu pro klasifikaci dopravních značek

```
keras_model.save('/content/gdrive/My Drive/DMP_Neuronove_site/keras_gtsrb_model.h5')
```