# Projekt: Klasifikácia zvierat pomocou umelej inteligencie (ResNet50)

## Úvod

Tento projekt je zameraný na klasifikáciu obrázkov zvierat pomocou umelej inteligencie. Použijeme predtrénovaný model ResNet50, ktorý sa doučí na dátach zvierat. Študenti budú pracovať s obrázkami zvierat, rozdeliť ich na tréningové, validačné a testovacie dáta a natrénovať model.

## Dataset

1. Stiahnite si dataset zvierat z Kaggle (alebo iného dostupného zdroja) – odporúčaný dataset obsahuje 90 druhov zvierat.  
2. Vložte priečinok `animals/` a súbor `labels.csv` do projektu.  
3. Dataset bude obsahovať obrázky rozdelené podľa priečinkov podľa druhu zvieraťa.

## Postup

1. Importujte potrebné knižnice (torch, torchvision, pandas, numpy, matplotlib, PIL).
2. Vytvorte triedu AnimalDataset, ktorá bude načítavať obrázky a ich kategórie z labels.csv.
3. Pomocou transformácií upravte všetky obrázky do rovnakého rozmeru (224x224) a normalizujte ich.
4. Rozdeľte dataset na 3 časti: trénovanie (80%), validácia (10%), testovanie (10%).
5. Načítajte predtrénovaný model ResNet50 a zmeňte jeho poslednú (výstupnú) vrstvu na počet vašich kategórií (napr. 11).
6. Natrénujte model na trénovacej množine a sledujte presnosť na validačnej množine počas 10 epoch.
7. Vyhodnoťte presnosť na testovacej množine a zobrazte graf vývoja presnosti a chyby počas učenia.
8. Vytvorte funkciu, ktorá náhodne vyberie obrázok z testovacej množiny a model predpovie, o aké zviera ide.

## Čo odovzdať

Študent má odovzdať Jupyter notebook (.ipynb alebo .py), v ktorom bude:  
- načítanie datasetu,  
- kód pre spracovanie a trénovanie modelu,  
- graf učenia (presnosť a strata),  
- výpis predikcií modelu na niekoľkých náhodných obrázkoch.

## Tipy

- Nezabudnite nastaviť správne cesty k datasetu.  
- Použite GPU ak je dostupné (`torch.device('cuda' if torch.cuda.is\_available() else 'cpu')`).  
- Ak sa trénovanie nedarí, skontrolujte správnosť kategórií v `labels.csv`.  
  
  
**Postup :**

1.Priprav priečinky a súbory:

projekt/

├─ data/

│ ├─ animals/ # koreň obrázkov (podpriečinky/obrázky, alebo ľubovoľná štruktúra)

│ └─ labels.csv # mapovanie: image\_name,category

└─ train\_animals\_resnet.py # kód nižšie

2. labels.csv – formát (príklad):

image\_name,category

cat/0001.jpg,Cat

cat/0002.jpg,Cat

dog/0001.jpg,Dog

eagle/001.png,Eagle

 image\_name je **cesta relatívna** k data/animals/ (podpriečinok/súbor).

 category je textový názov triedy (napr. „Cat“, „Dog“…).

**1) Kompletný kód (vložiť do train\_animals\_resnet.py)**

# =========================

# KROK 1: Import knižníc

# =========================

import os, random, time, sys

import numpy as np

import pandas as pd

import torch

import torch.nn as nn

from torch.utils.data import Dataset, DataLoader, random\_split

from PIL import Image

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

from torchvision import transforms as T

from torchvision.models import resnet50, ResNet50\_Weights

# Bezpečné nastavenia (reprodukovateľnosť)

SEED = 42

random.seed(SEED); np.random.seed(SEED); torch.manual\_seed(SEED)

device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")

print("Používam:", device)

# Cesty k dátam (uprav len ak máš inú štruktúru)

ROOT\_DIR = os.path.dirname(os.path.abspath(\_\_file\_\_))

DATA\_DIR = os.path.join(ROOT\_DIR, "data", "animals")

LABELS\_CSV = os.path.join(ROOT\_DIR, "data", "labels.csv")

# =========================

# KROK 2: Dataset trieda

# =========================

class AnimalDataset(Dataset):

"""

Načíta obrázky podľa labels.csv. Očakáva dva stĺpce:

- image\_name (relatívna cesta k DATA\_DIR)

- category (textový názov triedy)

"""

def \_\_init\_\_(self, root\_dir, labels\_df, transform=None):

self.root\_dir = root\_dir

self.transform = transform

self.labels\_df = labels\_df.reset\_index(drop=True) # pre istotu

# Vytvor mapovanie image -> encoded\_label

self.paths = self.labels\_df["image\_name"].tolist()

self.labels = self.labels\_df["category\_enc"].tolist()

def \_\_len\_\_(self):

return len(self.paths)

def \_\_getitem\_\_(self, idx):

rel\_path = self.paths[idx]

full\_path = os.path.join(self.root\_dir, rel\_path)

img = Image.open(full\_path).convert("RGB")

if self.transform:

img = self.transform(img)

label = int(self.labels[idx]) # ako LongTensor

return img, label

# ==========================================

# KROK 3: Transformácie (resize + normaliz.)

# ==========================================

transform = T.Compose([

T.Resize((224, 224)),

T.RandomHorizontalFlip(p=0.5),

T.ToTensor(),

T.Normalize(mean=[0.485, 0.456, 0.406],

std=[0.229, 0.224, 0.225]),

])

# ==========================================

# Načítanie labels.csv a enkód kategórií

# ==========================================

labels\_df = pd.read\_csv(LABELS\_CSV)

# enkóduj textové kategórie na čísla

encoder = LabelEncoder()

labels\_df["category\_enc"] = encoder.fit\_transform(labels\_df["category"])

class\_names = list(encoder.classes\_)

num\_classes = len(class\_names)

print("Triedy:", class\_names, "| num\_classes:", num\_classes)

# Dataset

full\_dataset = AnimalDataset(DATA\_DIR, labels\_df, transform=transform)

# ===================================================

# KROK 4: Rozdelenie na train/val/test = 80/10/10

# ===================================================

n = len(full\_dataset)

train\_size = int(0.8 \* n)

val\_size = int(0.1 \* n)

test\_size = n - train\_size - val\_size

train\_ds, val\_ds, test\_ds = random\_split(full\_dataset, [train\_size, val\_size, test\_size],

generator=torch.Generator().manual\_seed(SEED))

batch\_size = 32

train\_loader = DataLoader(train\_ds, batch\_size=batch\_size, shuffle=True, drop\_last=True)

val\_loader = DataLoader(val\_ds, batch\_size=batch\_size, shuffle=False, drop\_last=False)

test\_loader = DataLoader(test\_ds, batch\_size=batch\_size, shuffle=False, drop\_last=False)

print(f"Počty: train={len(train\_ds)}, val={len(val\_ds)}, test={len(test\_ds)}")

# ==========================================================

# KROK 5: Predtrénovaný ResNet50 + zmena poslednej vrstvy

# ==========================================================

weights = ResNet50\_Weights.DEFAULT

model = resnet50(weights=weights)

# Zamrazíme „telo“ a trénujeme len finálnu vrstvu (ľahšie pre SŠ)

for p in model.parameters():

p.requires\_grad = False

# Nahradíme výstupnú vrstvu na náš počet tried

model.fc = nn.Linear(model.fc.in\_features, num\_classes)

model = model.to(device)

# ==================================================

# KROK 6: Tréning – 10 epôch, sledujeme val presnosť

# ==================================================

criterion = nn.CrossEntropyLoss()

optimizer = torch.optim.Adam(model.fc.parameters(), lr=1e-3, weight\_decay=1e-4)

EPOCHS = 10

history = {

"train\_loss": [], "val\_loss": [],

"train\_acc": [], "val\_acc": []

}

def epoch\_pass(loader, train=False):

if train:

model.train()

else:

model.eval()

total\_loss, correct, total = 0.0, 0, 0

with torch.set\_grad\_enabled(train):

for X, y in loader:

X, y = X.to(device), y.to(device)

logits = model(X)

loss = criterion(logits, y)

if train:

optimizer.zero\_grad()

loss.backward()

optimizer.step()

total\_loss += loss.item() \* X.size(0)

preds = logits.argmax(1)

correct += (preds == y).sum().item()

total += X.size(0)

avg\_loss = total\_loss / total if total else 0.0

acc = 100.0 \* correct / total if total else 0.0

return avg\_loss, acc

start = time.time()

for epoch in range(1, EPOCHS+1):

tr\_loss, tr\_acc = epoch\_pass(train\_loader, train=True)

val\_loss, val\_acc = epoch\_pass(val\_loader, train=False)

history["train\_loss"].append(tr\_loss)

history["val\_loss"].append(val\_loss)

history["train\_acc"].append(tr\_acc)

history["val\_acc"].append(val\_acc)

print(f"Epoch {epoch:02d}/{EPOCHS} | "

f"train\_loss={tr\_loss:.4f} val\_loss={val\_loss:.4f} | "

f"train\_acc={tr\_acc:.1f}% val\_acc={val\_acc:.1f}%")

print(f"Hotovo za {time.time()-start:.1f}s")

# ======================================================

# KROK 7: Test presnosť + grafy (loss/accuracy vývoj)

# ======================================================

test\_loss, test\_acc = epoch\_pass(test\_loader, train=False)

print(f"TEST: loss={test\_loss:.4f} | acc={test\_acc:.1f}%")

plt.figure(figsize=(12,4))

plt.subplot(1,2,1)

plt.plot(history["train\_loss"], label="train")

plt.plot(history["val\_loss"], label="val")

plt.title("Loss")

plt.xlabel("Epoch"); plt.ylabel("CE loss"); plt.legend(); plt.grid(True)

plt.subplot(1,2,2)

plt.plot(history["train\_acc"], label="train")

plt.plot(history["val\_acc"], label="val")

plt.title("Accuracy")

plt.xlabel("Epoch"); plt.ylabel("Accuracy %"); plt.legend(); plt.grid(True)

plt.tight\_layout()

plt.show()

# Uloženie modelu (voliteľné)

torch.save({"model\_state": model.state\_dict(),

"classes": class\_names}, os.path.join(ROOT\_DIR, "animals\_resnet50.pth"))

# ======================================================

# KROK 8: Predikcia náhodného obrázka z testu

# ======================================================

# Pomocná denormalizácia na pekné zobrazenie

def denorm(img\_tensor):

# inverse normalization (ImageNet)

mean = np.array([0.485, 0.456, 0.406])

std = np.array([0.229, 0.224, 0.225])

img = img\_tensor.detach().cpu().numpy().transpose(1,2,0)

img = (img \* std) + mean

return np.clip(img, 0, 1)

def predict\_random\_from\_test(kusov=5):

model.eval()

# prístup k pôvodným indexom v Subsete

idxs = random.sample(range(len(test\_ds)), k=min(kusov, len(test\_ds)))

plt.figure(figsize=(4\*kusov, 4))

for p, i in enumerate(idxs, start=1):

img, y\_true = test\_ds[i]

with torch.no\_grad():

logits = model(img.unsqueeze(0).to(device))

pred\_id = int(logits.argmax(1).item())

y\_name = class\_names[y\_true]

p\_name = class\_names[pred\_id]

plt.subplot(1, kusov, p)

plt.imshow(denorm(img))

plt.axis("off")

plt.title(f"GT: {y\_name}\nPred: {p\_name}")

plt.tight\_layout()

plt.show()

# Spusti ukážku predikcií

predict\_random\_from\_test(kusov=5)

**Ako to spustiť (bez googlenia)**

1. Ulož kód do train\_animals\_resnet.py.
2. Skontroluj, že **data/labels.csv** aj **obrázky v data/animals/…** sedia (relatívne cesty v CSV).
3. V termináli projektu spusti:

python train\_animals\_resnet.py

Sleduj priebeh epôch v termináli, potom sa zobrazia dva grafy (loss/accuracy) a nakoniec okno s 5 náhodnými test obrázkami s predikciami.

**Poznámky / časté chyby**

* **FileNotFoundError** → nesedí cesta v labels.csv (stĺpec image\_name musí byť relatívny k data/animals/).
* **num\_classes** musí sedieť s počtom rôznych kategórií v labels.csv.
* Ak je dataset malý, prvé výsledky môžu byť slabšie — pridaj viac dát alebo zníž num\_workers (ak by robilo problémy na Windows).