

# Neurónové Siete

## Python Keras a Tensorflow



# Ako Začneme?

1. Pridajte si ma na **LinkedIn**

[www.linkedin.com/in/miroslav-reiter](http://www.linkedin.com/in/miroslav-reiter)

2. Stiahnite si Cvičný NTB Súbor

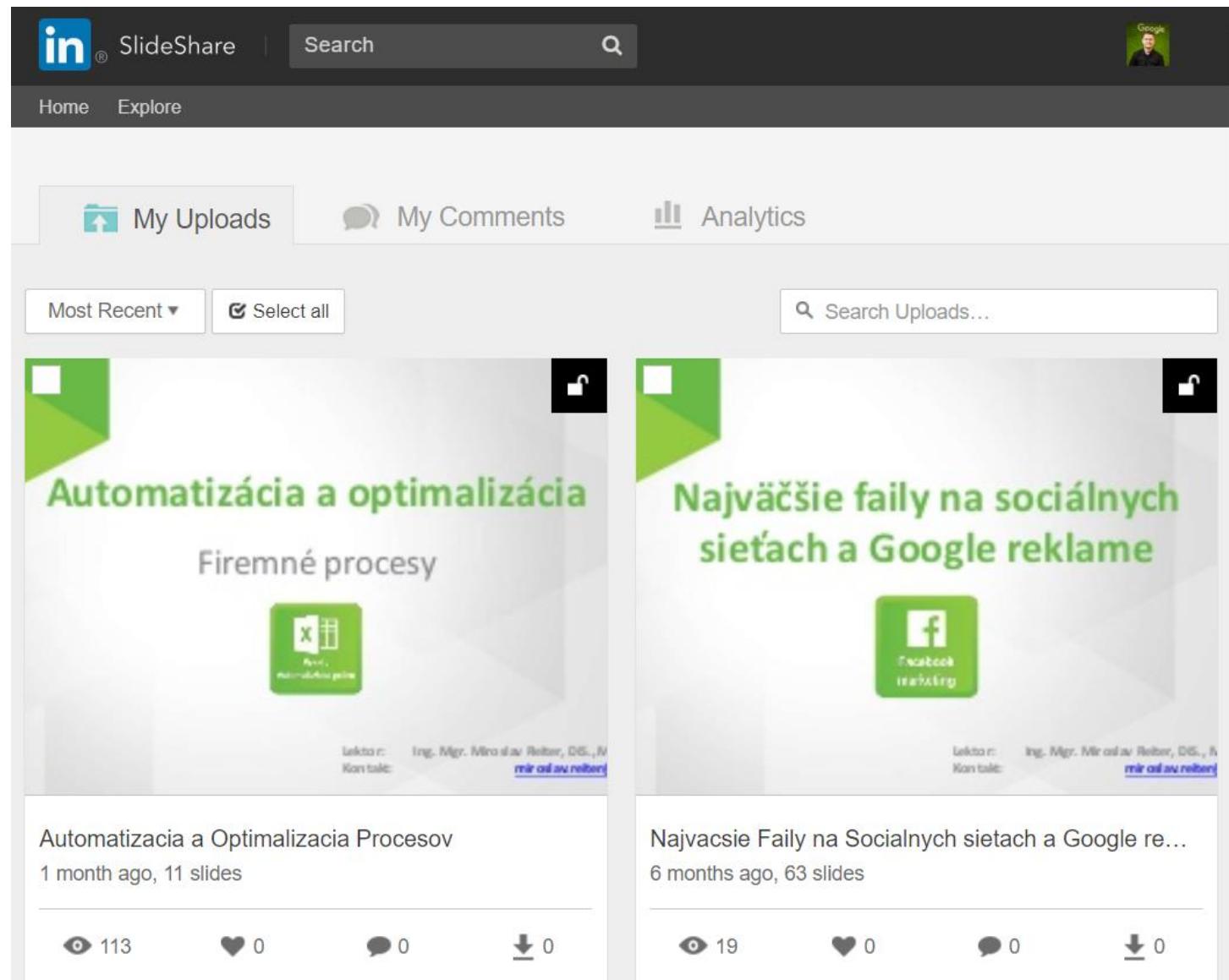
[github.com/miroslav-reiter/Kurzy\\_SAV\\_Analytika\\_Python\\_R](https://github.com/miroslav-reiter/Kurzy_SAV_Analytika_Python_R)

3. Prezentácia po prednáške

<https://github.com/miroslav-reiter/>

4. Videá na YouTube

[www.youtube.com/@VITA-Academy](http://www.youtube.com/@VITA-Academy)





miroslav-reiter

 Type / to search[Overview](#) [Repositories 108](#) [Projects](#) [Packages](#) [Stars 115](#) [Sponsoring](#)

## Miroslav Reiter

miroslav-reiter

Founder of VITA Academy ★ Microsoft Certified Trainer ★ Google, Android Certified Trainer ★ ISTQB Trainer

[Edit profile](#)

81 followers · 2 following

VITA Academy

Bratislava

miroslav.reiter@it-academy.sk

www.vita.sk

<https://orcid.org/0000-0003-1804-651X>

@VITA\_Academy\_SK

in/miroslav-reiter

VitaAcademySK

@VITA-Academy

### Achievements



miroslav-reiter / README.md

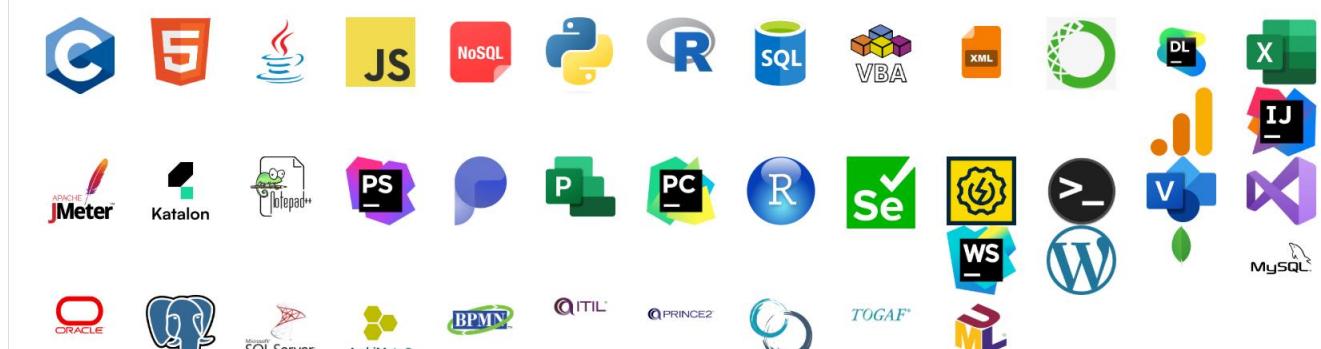
## About me

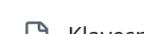
- 👉 Hi, I am @Miroslav-Reiter
- 👤 I am a Google Certified Trainer GCT, Microsoft Certified Trainer MCT, Microsoft Most Valuable Professional MVP, ISTQB/GASQB Trainer, PRINCE2/ITIL4/ArchiMate/TGAF/UML/BPMN/Scrum Trainer
- 🛠 My preferred programming languages are Java 🇪🇸, Python 🧑, JavaScript (Google Apps Script), VBA and R
- 👨‍🏫 I teach programming and how to use IT technologies effectively at [VITA](#) and [IT Academy](#). My online courses can be bought directly at [VITA.sk](#). I recommend buying an annual subscription with all courses  
Online Akreditované Kurzy a Skolenia [VITA](#)
- 📢 I frequently speak at conferences and workshops. Mostly about IT (programming, automation, certification, testing, enterprise architecture and modeling), management (Project management, PRINCE2, ITIL4), education and online marketing with ads (Google Ads, Google Analytics, Copywriting, Social media, YouTube)
- 🎥 I regularly upload a new videos to [YouTube channel Miroslav Reiter - VITA Academy](#)  
 [YouTube IT Academy](#) 9.5k
- ⚡ Fun fact: I have graduated from 14 different Universities. I am currently studying at 4 other Universities, 3 of which are PhDs. Check out my [LinkedIn](#) profile

## My Socials

 [Google](#) [VITA](#) [Google](#) [IT Academy](#)

## Languages, Tools and Frameworks





# Kurzy SAV DataScience, Python, R, Julia, BI, AI/ML, ChatGPT

Materiály, Zdrojové Kódy a Projekty, Prezentácie ku kurzom SAV DataScience, Python, OOP, R, BI, Analytika

Python je interpretovaný, interaktívny, open-source programovací jazyk. Python beží na mnohých variantoch Unixu, na Macu a Windowse (súčasťou kurzu bude inštalácia na vašom systéme). Pre absolvovanie kurzu je potrebné mať k dispozícii vlastný notebook (s ľubovoľným operačným systémom podporujúcim Python).

R je programovací jazyk a softvérové prostredie pre štatistickú analýzu a vizualizáciu. R je voľne dostupný pod GNU licenciou a existujú verzie pre mnohé operačné systémy ako Linux, Windows a Mac.

ChatGPT je pokročilý jazykový model umelej inteligencie vyvinutý spoločnosťou OpenAI. Je to variant modelu GPT (Generative Pre-trained Transformer), ktorá je špeciálne navrhnutá na generovanie textu a interakciu v dialógovej

## About



Materiály, Zdrojové Kódy, Prezentácie ku kurzom SAV Python, OOP, R, BI, Data Science

python jupyter numpy oop  
 jupyter-notebook pandas python3  
 matplotlib sav beautifulsoup reiter  
 matplotlib-pyplot

Readme

Activity

22 stars

16 watching

23 forks

## Releases

No releases published

[Create a new release](#)

## Packages

No packages published

[Publish your first package](#)

## Languages



## Suggested workflows

 PEVS-PANI-Kontrolling Public[Pin](#) [Unwatch 1](#) [Fork 1](#) [Starred 1](#) main 1 Branch 0 Tags Go to file Add file Code

## About



 Zdrojové kódy a projekty z predmetu Kontrolling na DNN na PEVS/PANI

 python  deep-learning  tensorflow  
 keras  dnn  uns  reiter  pani  
 pevs  kontrolling

 Readme Activity 1 star 1 watching 1 fork

## Releases

No releases published

[Create a new release](#)

## Packages

No packages published

[Publish your first package](#)

## Languages

 Jupyter Notebook 99.7%  Python 0.3%

## Suggested workflows

Based on your tech stack

 Python application [Configure](#)  
Create and test a Python application.

 Django [Configure](#)  
Build and Test a Django Project

 README Kontrolling (PEVS-PANI)

 Zdrojové kódy a projekty z predmetu Kontrolling na DNN (Deep Neural Network) na PEVS/PANI

## Trénovacie a Validačné Vzorky

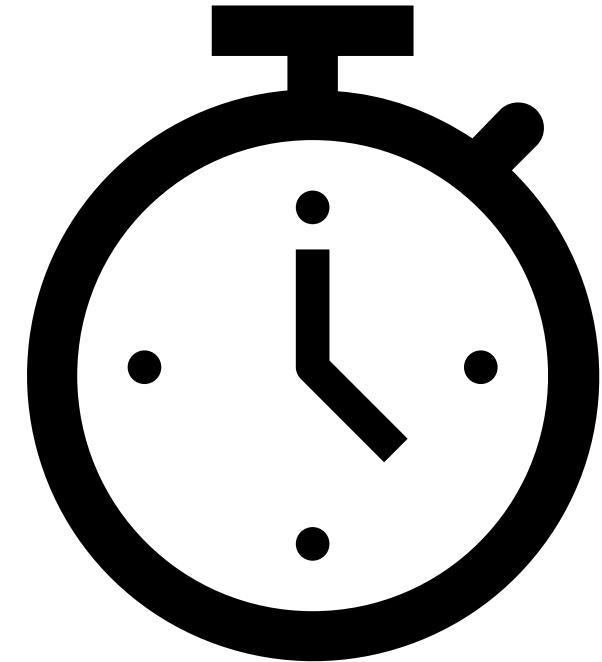
Na vstupe (vstupnej vrstve) máme RGB obrázky o veľkosti 128 x 128 px, 3 kanály. Veľkosť trénovacích vzoriek je 1098 a počet validačných vzoriek je 277. Rozdelenie je teda 80 % trénovacie vzorky a 20 % validačné t.j. paretovo pravidlo 80/20.

Celkový počet vzoriek:	1370	100%
Počet trénovacích vzoriek:	1098	80%
Počet validačných vzoriek:	272	20%

## Použité experimenty

# Úvodné Informácie

- Časový rozvrh (9:00-13:30)
  - Programujeme (50 min)
  - Prestávky (10 min)
  - Obedová prestávka
  - Mobilné telefóny a zariadenia
- 
- Priprav si otázky a rovno sa pýtaj
  - Interaktívna forma



# O mne - Miroslav Reiter

7

**40000+  
klientov a  
1000+ firiem**

**IT Architekt  
Programátor  
Manažér**

**Microsoft  
Google  
ISTQB tréner**

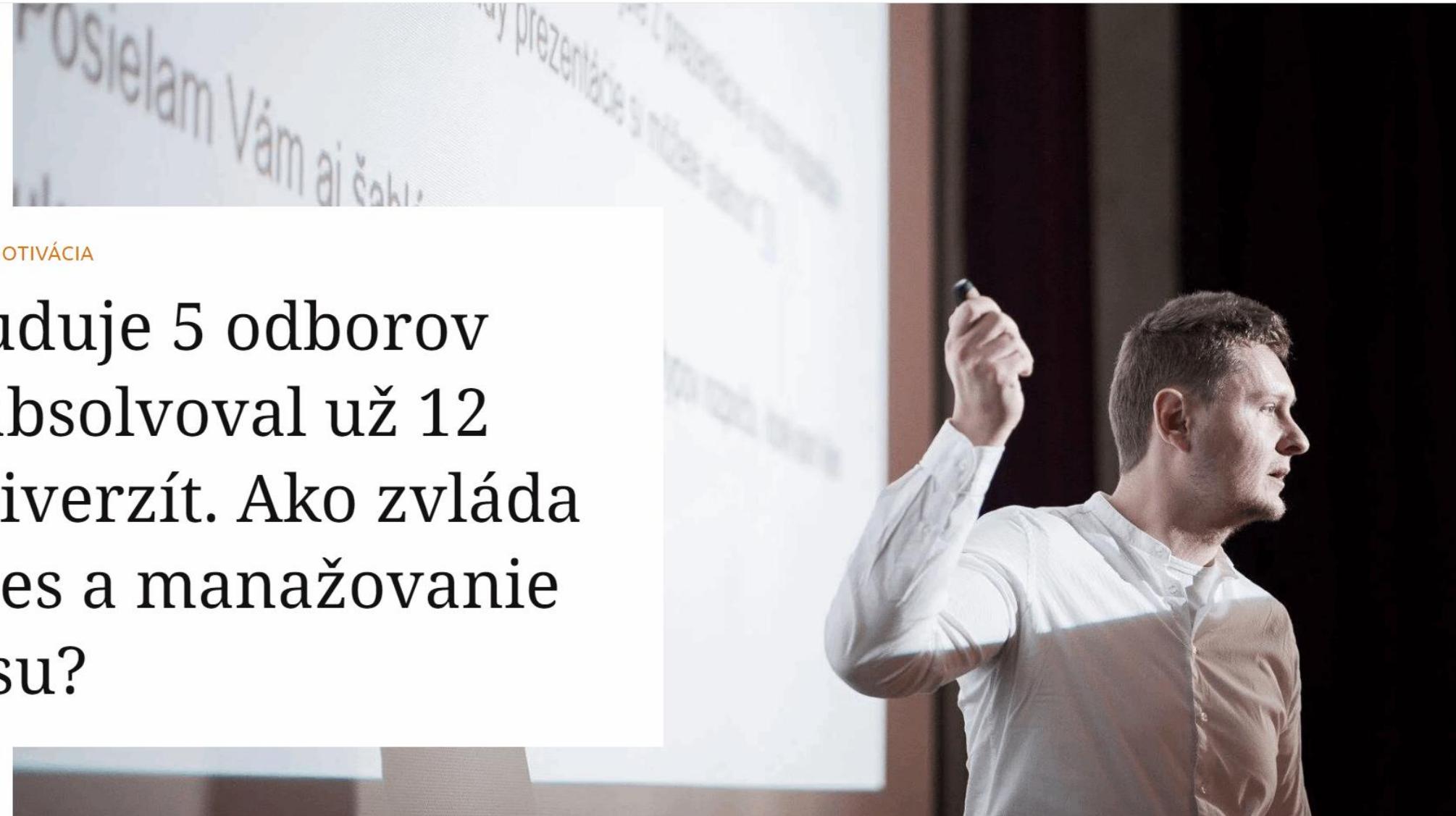
**134 certifikácií**

**151 príručiek a  
publikácií**

**14 škôl**

**62 projektov**

**Vlastná firma**



---

**MOTIVÁCIA**

# Študuje 5 odborov a absolvoval už 12 univerzít. Ako zvláda stres a manažovanie času?

Foto: Jakub Kovalík pre FMK UCM | Miroslav Reiter na prednáške Grow with Google na FMK UCM.



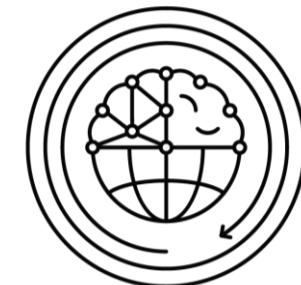
**Nikola Kotláriková**  
19. júl 2022 · 8 min. čítania



# Miroslav Reiter



1. PhDr. VŠM (Podnikovný manažment)
2. Ing. STU FEI (**Aplikovaná informatika**)
3. Mgr. UK FM (**Strategický manažment a marketing**)
4. Mgr. VŠM (**Manažérstvo kvality**)
5. Mgr. VŠEMVŠ (Verejná správa)
6. Mgr. DTI (**Učiteľstvo ekonomických predmetov**)
7. DiS. AMOS (Cestovný ruch)
8. **MBA LIGS (Executive management)**
9. **DBA Humanum (IT manažment)**
10. MPA IES (Verejná správa a samospráva krajov)
11. MSc. Humanum (**Bezpečnosť inf. systémov**)
12. Ing. Paed. IGIP STU
13. Mgr. PEVŠ (**Bezpečnosť informačných systémov**)
14. RNDR. PEVŠ (**Bezpečnosť informačných systémov**)



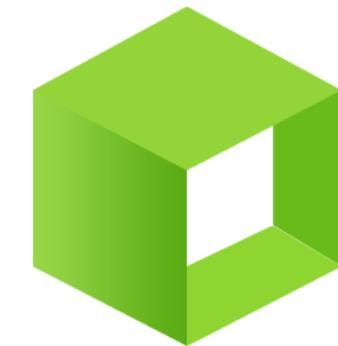
**FAKULTA MANAGEMENTU**  
Univerzita Komenského  
v Bratislave

**DIGITÁLNA  
UNIVERZITA**

STU  
FIIT



 **VITA**  
ACADEMY



**IT ACADEMY**

## Most Valuable Professionals

[About](#) [Events](#) [Find an MVP](#)[Profile](#) [Events](#)

### Headline

 IT Architect and Programmer  Hard worker   
Lecturer and Certified Trainer

### Biography

 I'm a hard worker, intellectual and joker. I love learning and teaching as well. My main objective is to teach people and improve their IT knowledge. To create truly practical knowledge necessary for life and present it in an interesting way. I don't like snake charmers and people who cannot...

[▼ Read more](#)

## Miroslav Reiter

 Slovakia IT Academy s.r.o.

### Most Valuable Professionals

## High Impact Activities

This community leader has not added a high impact activity yet.

### Award Category

M365

### Technology Area

Visio, Excel

### Languages

English, Slovak

### Social



 [Domov](#) > [Registre](#) > [Znalcí](#) > PhDr. Ing. Miroslav Reiter

Znalec

# PhDr. Ing. Mgr. Miroslav Reiter

Evidenčné číslo: 915864

## Miesto výkonu činnosti

Tomášikova 50  
83104 Bratislava  
Slovenská republika

[Zobraziť na mape](#)

## Kontaktné údaje

Mobil: +421 908 163 084  
E-mail: [znapec@it-academy.sk](mailto:znapec@it-academy.sk)

## Odbory a odvetvia

Odbor / Odvetvie	História	Stav
<b>100000 - Elektrotechnika</b>		
100400 - Riadiaca technika, výpočtová technika (hardware)	14.02.2023 - <b>Zápis</b>	<b>AKTÍVNY</b>
100800 - Nosiče zvukových a zvukovoobrazových záznamov	14.02.2023 - <b>Zápis</b>	<b>AKTÍVNY</b>
100900 - Počítačové programy (software)	14.02.2023 - <b>Zápis</b>	<b>AKTÍVNY</b>
101000 - Bezpečnosť a ochrana informačných systémov	14.02.2023 - <b>Zápis</b>	<b>AKTÍVNY</b>

# Vyberte si online kurz

Naučte sa programovať, tvoriť webstránky a grafiku, manažovať alebo sa zamerajte na osobný rozvoj. Všetko jednoducho vďaka našim online kurzom z pohodlia domova.

## Ročné Predplatné na všetky Online Kurzy

2290 €

490 €

Prístup pre Vás do všetkých Aktuálnych aj  
Pripravovaných Online Kurzov

12 mesačná platnosť

 Kúpiť teraz

Zistiť viac



560 kurzov v ponuke



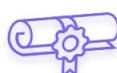
Zábavné online lekcie



Akreditované kurzy



13 rokov skúseností



Certifikovaní profesionálni lektori

Odporúčame Kurzy špeciálne pre vás



Online kurz SAP I.  
Začiatočník  
~~298,00€~~ 398,00€



Balík SAP Profesionál  
~~998,80€~~ 998,80€



Online kurz ChatGPT a DALL-E AI  
~~186,00€~~ 254,00€



Online kurz SQL I.  
Začiatočník  
~~196,00€~~ 276,00€



Program MPA Manažér Štátnej a Verejnej Správy  
~~1 940,00€~~ 2 540,00€



Online kurz Lektor (Akreditovaný Kurz Lektor)  
~~238,00€~~ 296,00€

Chat



Všetko Videá Obrázky Správy Krátke videá Knihy Web :: Viac

## Nástroje

[www.youtube.com › watch](http://www.youtube.com/watch)

Ako začať programovať v Pythone? - Online kurz Python a ...



Akreditovaný online kurz Python I. Začiatočník: →  
<https://www.vita.sk/online-kurz-python-i-zaciatocnik/> Akreditovaný prezenč...

YouTube · Miroslav Reiter - VITA Academy · 10. 11. 2021

16 kľúčových momentov v tomto videu

[www.youtube.com/watch](http://www.youtube.com/watch)

Online Kurz Python I. Začiatočník - Ukážka - IDE, Pycharm ...



Online Kurz Python I. Začiatočník - Ukážka - IDE, Pycharm, Print, Premenné, Konverzie, Netbeans · Comments6.

YouTube · Miroslav Reiter - VITA Academy · 2. 3. 2020

10 kľúčových momentov v tomto videu

[www.youtube.com/watch](http://www.youtube.com/watch?v=...)

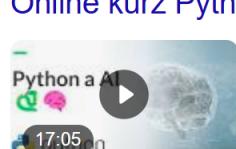


Prednášajúci: PhDr. Ing. Mgr. et Mgr. et Mgr. et Mgr. Miroslav Reiter, DiS.,  
MBA, MPA, MSC, DBA, Ing., Paed., IIGIP. Miesto: SAV v Bratislave.

YouTube : Miroslav Reiter - VITA Academy ; 25.8.2023

4 klúčové momenty v tomto video

[www.youtube.com/watch](http://www.youtube.com/watch)



[... Online kurz Python - Používanie AI pri Programovaní v Pythone a Generovanie Zdrojových Kódov Akreditovaný online kurz Programovanie](#)

YouTube : Miroslav Reiter - VITA Academy ; 25. 9. 2023

4 klíčové my

4 Kræske momenty v tomto video



Comments 4 ; Online kurz Python - Cyklus While, break a continue. Miroslav Beiter, VITA Academy ; 156 views ; Online kurz Jazyk Julia Ako

[Všetko](#) [Obrázky](#) [Videá](#) [Správy](#) [Knihy](#) [Finance](#) [Krátke videá](#) [:: Viac](#)[Nástroje](#)

www.youtube.com › watch

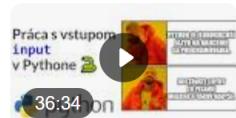
[Ako začať programovať v Pythone? - Online kurz Python a ...](#)

... Python I. Začiatočník: → <https://www.it-academy.sk/kurz/python/> Online kurz Python ... Online #Kurzy #Školenia #Programovanie #Python...

YouTube · Miroslav Reiter - VITA Academy · 10. 11. 2021

16 kľúčových momentov v tomto videu ▾

www.youtube.com › watch

[Online kurz Python Základy - Práca s Vstupom \(input\) v ...](#)

... Online #Kurzy #Školenia #Programovanie #Jetbrains #Datalore #Jupyter.  
... <https://www.vita.sk/on...> Akreditovaný online kurz Programovanie ...

YouTube · Miroslav Reiter - VITA Academy · 25. 8. 2023

4 kľúčové momenty v tomto videu ▾

www.youtube.com › watch

[Online Kurz Python I. Začiatočník - Ukážka - IDE, Pycharm ...](#)

... kurz/python/ Online kurz Python I. Začiatočník je pre teba vhodný, ak si ...  
... Online #Kurzy #Školenia #Programovanie #Jazyk #Python #Idle.

YouTube · Miroslav Reiter - VITA Academy · 2. 3. 2020

10 kľúčových momentov v tomto videu ▾

www.youtube.com › watch

[Online kurz Python - Používanie AI pri Programovaní v ...](#)

... Online kurz Python - Používanie AI pri Programovaní v Python a Generovanie Zdrojových Kódov Akreditovaný online kurz Programovanie...

YouTube · Miroslav Reiter - VITA Academy · 25. 9. 2023

4 kľúčové momenty v tomto videu ▾

www.youtube.com › watch

[Online kurz Python - Čo je to Python? - Na čo a kde sa Python ...](#)

Comments3 · Online Kurz Python Objektové Programovanie - Čo je to Polymorfizmus v Pythone? · Online Kurz Úvod do Programovania - Ukážk...



Luigi, Mário  
a Yoshi

# Čo Robíte?

1. Študent/Učiteľ

2. Zamestnanec

3. Podnikateľ

4. Nezamestnaný/materská

5. Dievča pre všetko



# Vaše Ciele a Očakávania

1. Doplniť si znalosti z jazyka Python

2. Naučiť sa knižnicu Keras (TensorFlow)

3. Vytvoriť a natrénovať jednoduchý model NS

4. Pochopit štruktúru NS v Keras

5. Práca s modelmi a výsledkami

Zábava je v zaručená v každom bode 😊



# Online Kurz Python Neurónové siete s knižnicou Keras

Vitajte v našom kurze zameranom na neurónové siete a knižnicu Keras v jazyku Python. Naučíte sa základy neurónových sietí, ich implementáciu a optimalizáciu.

Čo sa naučíte:

- Základné koncepty strojového učenia a neurónových sietí
- Inštalácia a konfigurácia knižnice Keras
- Tvorba a trénovanie vlastných neurónových modelov
- Riešenie praktických problémov pomocou hlbokého učenia

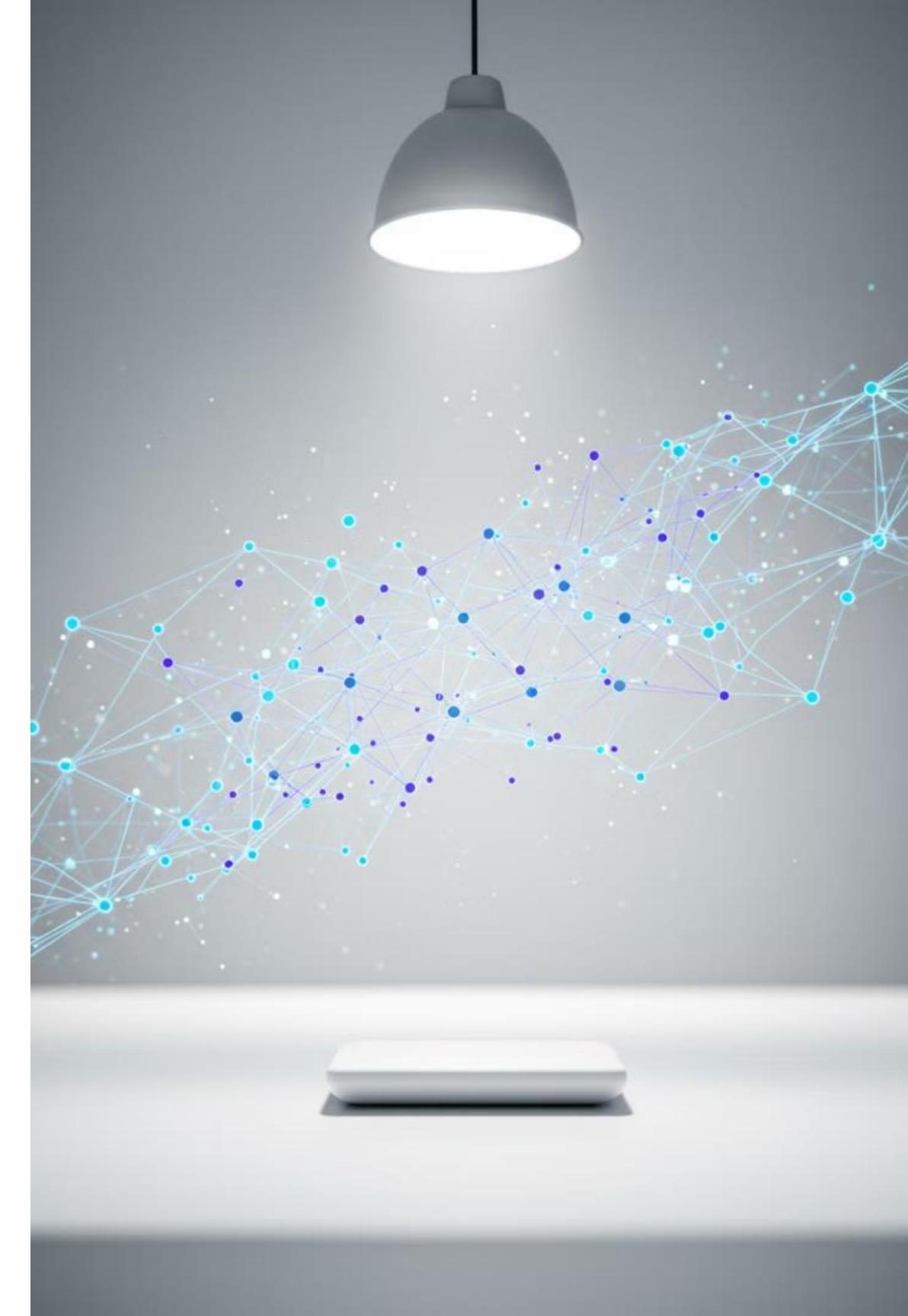
Pre koho je kurz určený:

Kurz je vhodný pre začiatočníkov aj pokročilých programátorov so základnou znalosťou Pythonu, ktorí sa chcú ponoriť do sveta umelej inteligencie.

Technické požiadavky:

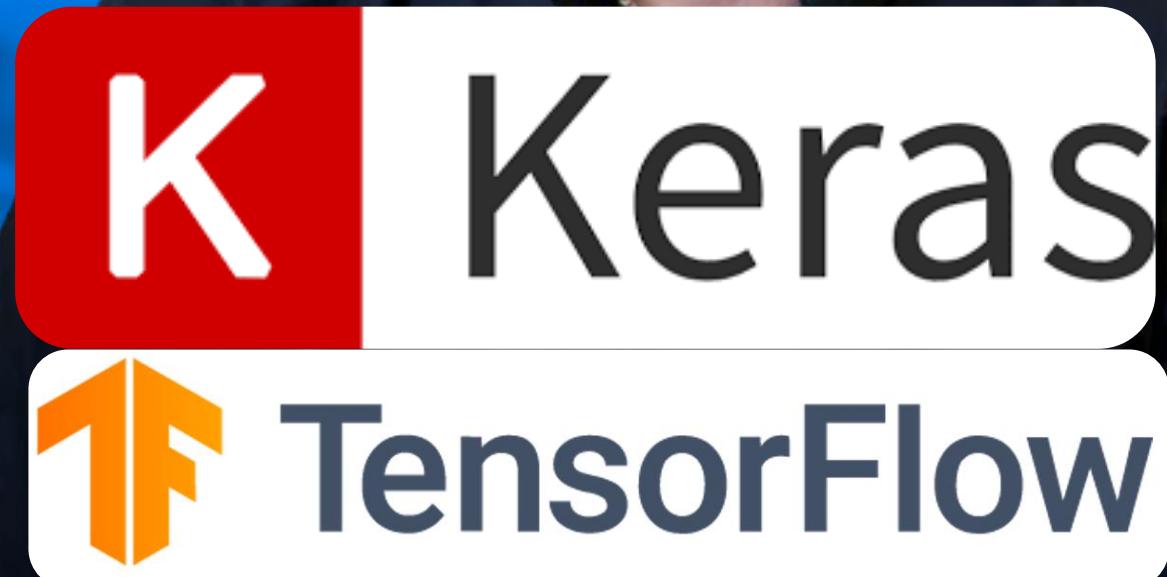
Python 3.8+, základné znalosti programovania, prístup k počítaču s pripojením na internet.

 podľa Miroslav Reiter



# Ako začať s Neuronovými Siet'ami

AKREDITOVANÝ KURZ

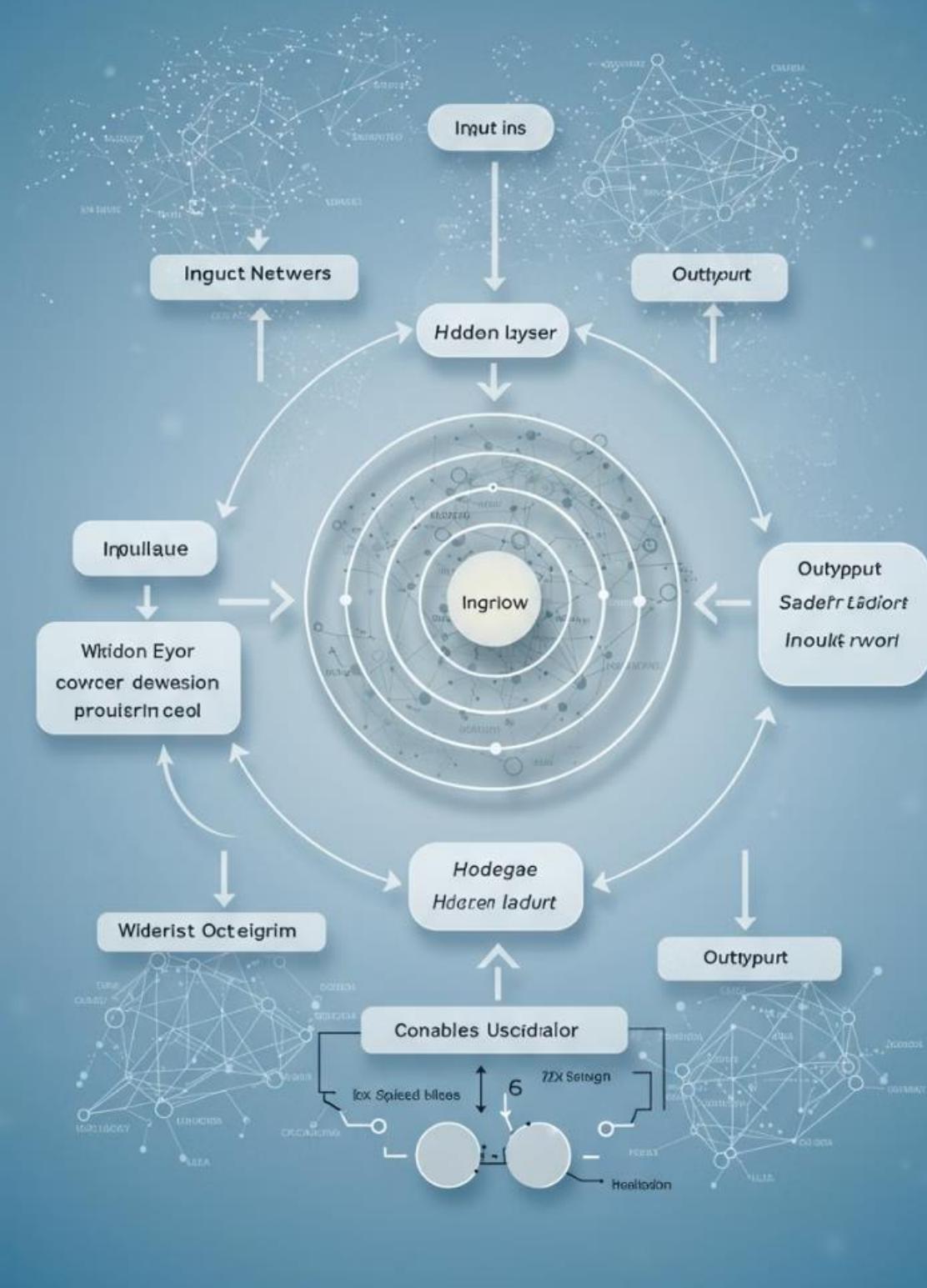




# Čo sa naučíme?

1. Čo sú to neurónové siete?
2. Z čoho sa skladá umelý neurón?
3. Aké sú najčastejšie typy aktivačných funkcií?
4. Čo je to Keras?
5. Na čo slúži metóda `compile()`?
6. Aký je rozdiel medzi Sequential a Functional API v Keras?
7. Profit

## Neural Network



# Čo sú neurónové siete?



## Biologická inšpirácia



Inšpirované biologickým  
neurónom v mozgu.  
Modelujú funkciu  $y = f(x)$ .



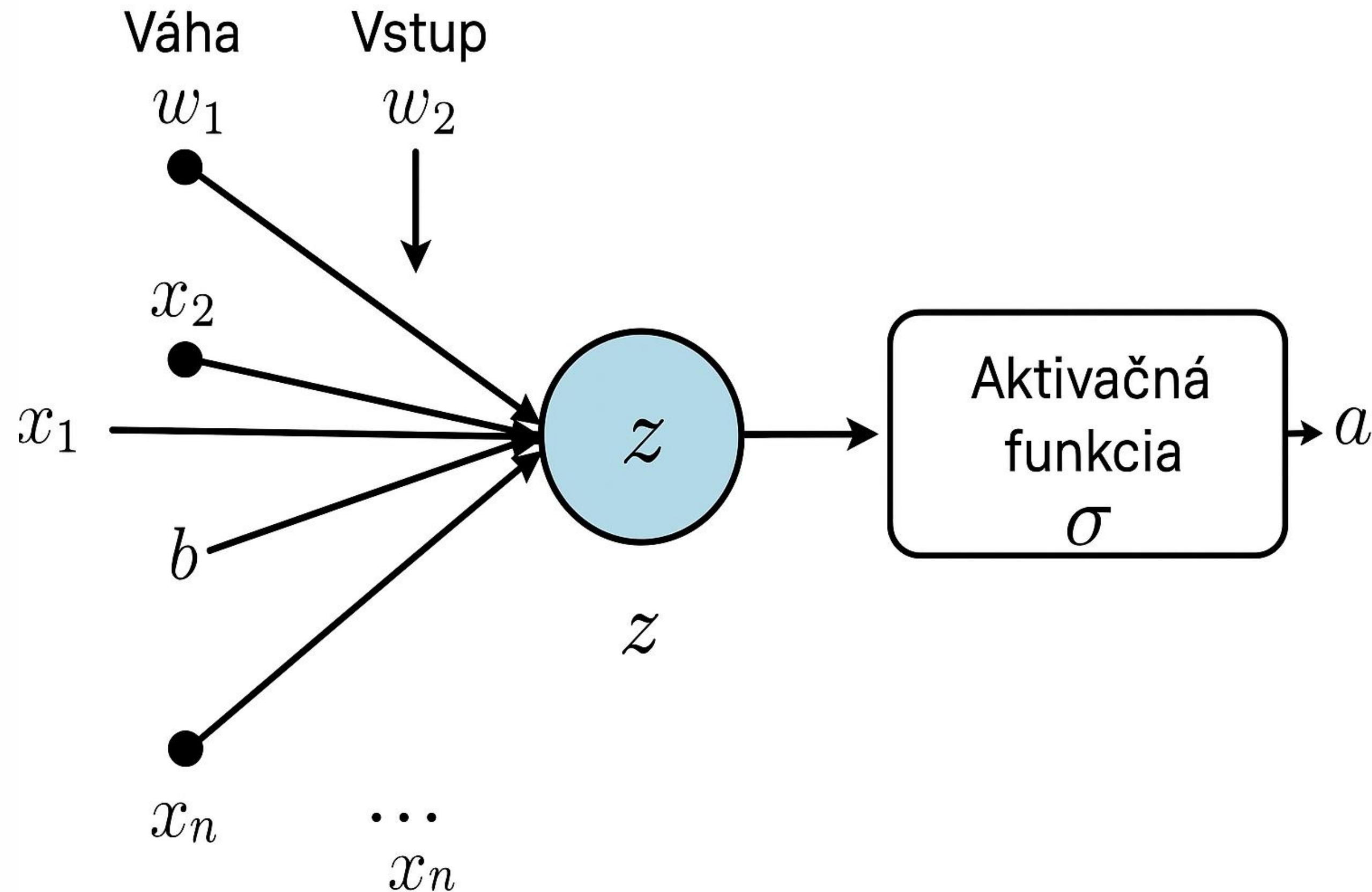
## Vrstvená štruktúra

Využívajú vstupnú vrstvu,  
skryté vrstvy a výstupnú  
vrstvu na spracovanie  
údajov.



## Učenie z chýb

Učia sa na základe chyby pomocou spätej propagácie  
(backpropagation).



# Matematický model neurónu



Matematicky vyjadrené:  $y = \varphi(\sum(w_i \cdot x_i) + b)$ , kde  $x_i$  sú vstupy,  $w_i$  sú váhy,  $b$  je bias a  $\varphi$  je aktivačná funkcia.

Tento základný model je stavebným kameňom komplexných neurónových sietí. Váhy a bias sú parametre, ktoré sa optimalizujú počas trénovalia, aby sa minimalizovala chyba predikcie.

# Aktivačné funkcie

## ReLU

$$\varphi(x) = \max(0, x)$$

Najpoužívanejšia aktivácia pre skryté vrstvy. Rýchla konvergencia.

Výhodou je jednoduchý výpočet a odstránenie problému miznúceho gradientu. Nevýhodou je problém "dying ReLU" pri negatívnych vstupoch.

Volba vhodnej aktivačnej funkcie výrazne ovplyvňuje rýchlosť učenia a celkovú výkonnosť neurónovej siete pri riešení konkrétnych problémov.

## Sigmoid

$$\varphi(x) = 1/(1+e^{-x})$$

Výstup v rozmedzí 0-1. Vhodná pre binárnu klasifikáciu.

Historicky populárna, ale trpí problémom saturácie a miznúceho gradientu pri vysokých absolútnych hodnotách vstupu. Stále užitočná pre výstupné vrstvy.

## Softmax

$$\text{softmax}(z_i) = e^{z_i} / \sum e^{z_j}$$

Normalizovaný výstup ako pravdepodobnostná distribúcia pre multi-class klasifikáciu.

Generalizácia sigmoid funkcie pre viac tried. Súčet všetkých výstupov je vždy 1, čo umožňuje interpretáciu ako pravdepodobnosti príslušnosti k triedam.



# Typy neurónových sietí

## Perceptron

Najjednoduchší typ neurónovej siete s jednou vrstvou.

Inšpirovaný biologickým neurónom, má vstupy, váhy, aktivačnú funkciu a jeden výstup. Vhodný len pre lineárne separovateľné problémy.

## MLP (Multilayer Perceptron)

Sieť s viacerými vrstvami neurónov. Základný typ pre mnohé úlohy.

Obsahuje vstupnú vrstvu, jednu alebo viac skrytých vrstiev a výstupnú vrstvu. Dokáže riešiť aj nelineárne problémy vďaka aktivačným funkciám a viacerým vrstvám.

## CNN (Convolutional Neural Network)

Špecializované na spracovanie obrazu. Využíva konvolučné vrstvy.

Používa konvolučné filtre na rozpoznávanie lokálnych vzorov a pooling vrstvy na redukciu dimenzií. Vynikajúce v počítačovom videní, rozpoznávaní obrazov a klasifikácii.

## RNN (Recurrent Neural Network)

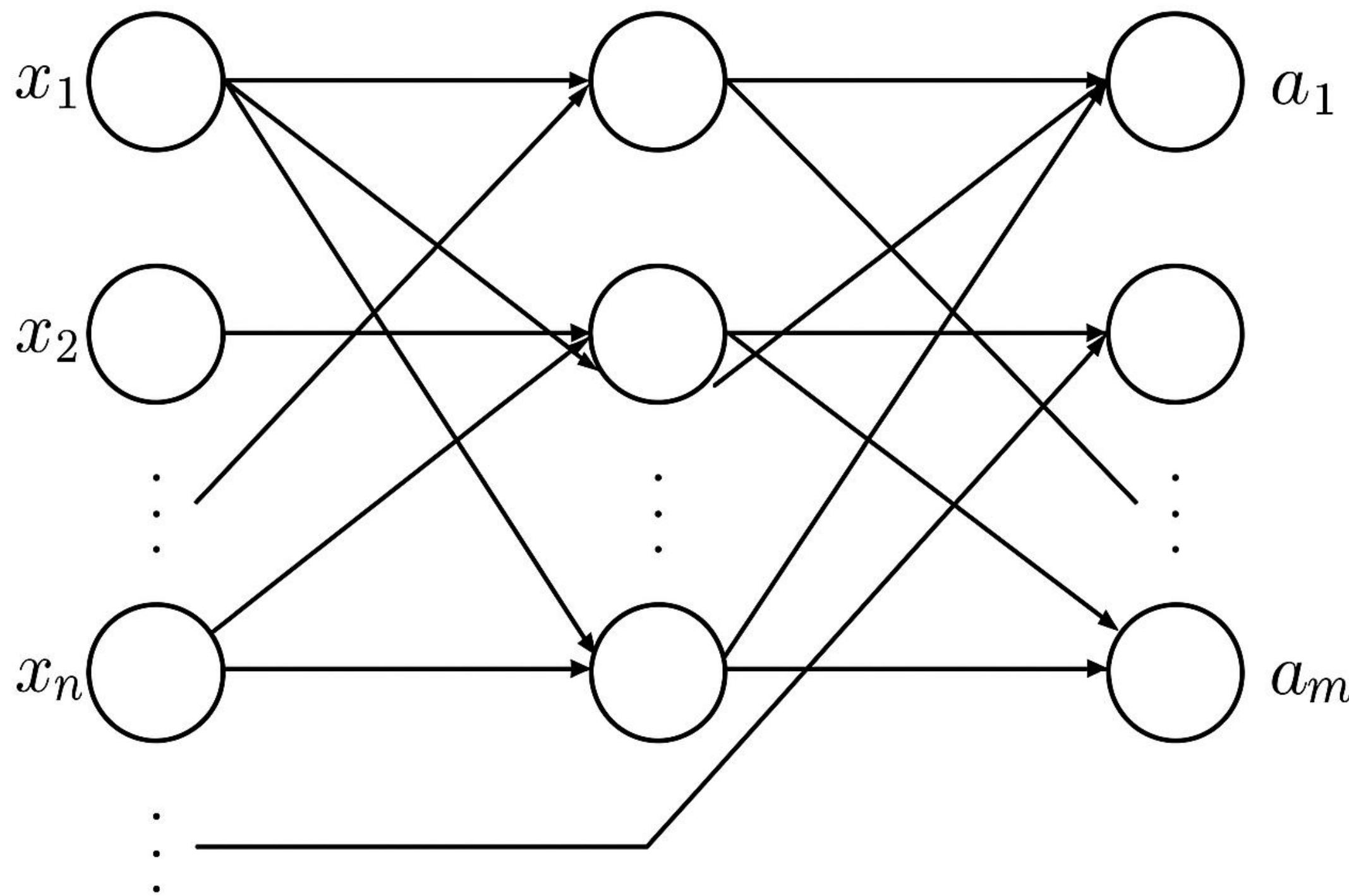
Vhodné pre sekvenčné dátá ako text alebo časové rady.

Majú spätnú väzbu, ktorá umožňuje uchovávať informácie z predchádzajúcich vstupov. Varianty ako LSTM a GRU riešia problém miznúceho gradientu pre dlhé sekvencie.

Input layer

Hidden layer

Output layer



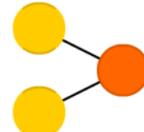
*A mostly complete chart of*

# Neural Networks

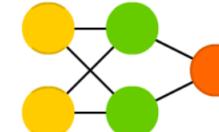
©2016 Fjodor van Veen - [asimovinstitute.org](http://asimovinstitute.org)

- (○) Backfed Input Cell
- (○) Input Cell
- (△) Noisy Input Cell
- (●) Hidden Cell
- (○) Probabilistic Hidden Cell
- (△) Spiking Hidden Cell
- (●) Output Cell
- (○) Match Input Output Cell
- (●) Recurrent Cell
- (○) Memory Cell
- (△) Different Memory Cell
- (●) Kernel
- (○) Convolution or Pool

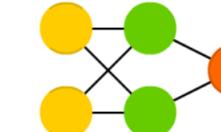
Perceptron (P)



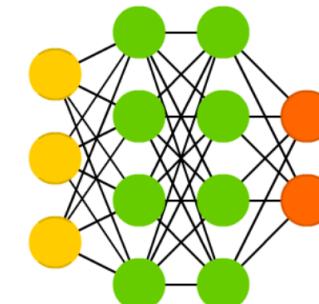
Feed Forward (FF)



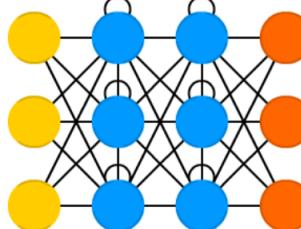
Radial Basis Network (RBF)



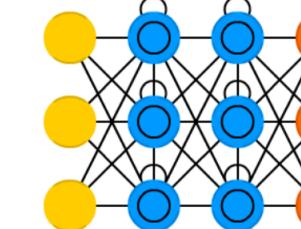
Deep Feed Forward (DFF)



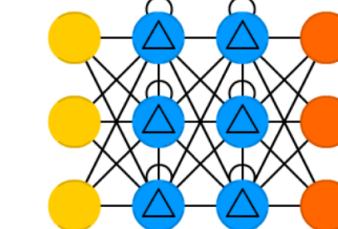
Recurrent Neural Network (RNN)



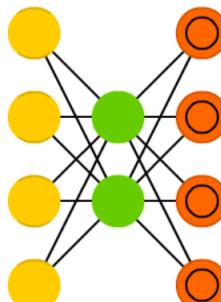
Long / Short Term Memory (LSTM)



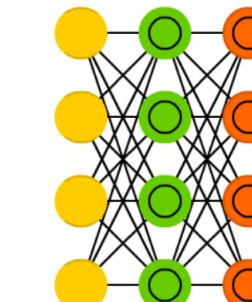
Gated Recurrent Unit (GRU)



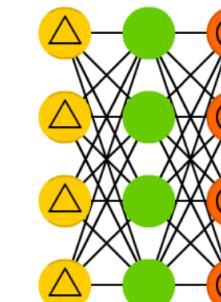
Auto Encoder (AE)



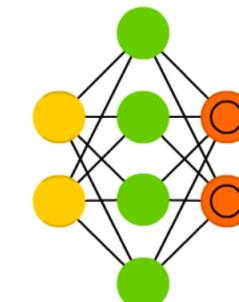
Variational AE (VAE)



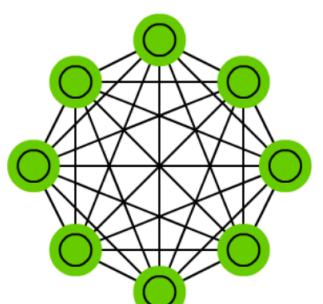
Denoising AE (DAE)



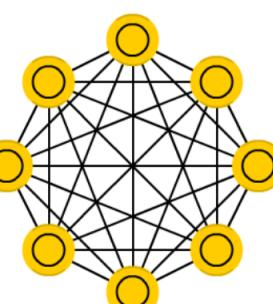
Sparse AE (SAE)



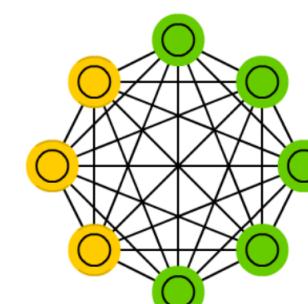
Markov Chain (MC)



Hopfield Network (HN)



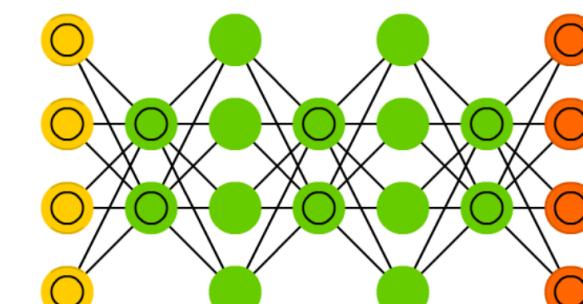
Boltzmann Machine (BM)



Restricted BM (RBM)

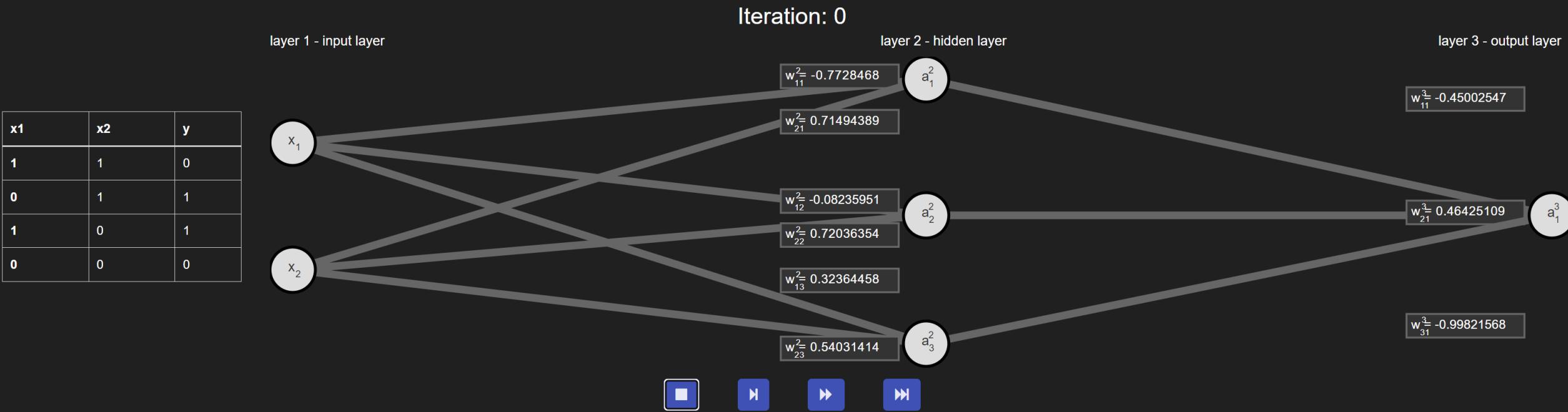


Deep Belief Network (DBN)



**NEURAL NETWORK SIMULATOR**

Neural Network Simulator is a real feedforward neural network running in your browser. The simulator will help you understand how artificial neural network works. The network is trained using backpropagation algorithm, and the goal of the training is to learn the XOR function. One forward and the backward pass of single training example is called iteration, each iteration consists of 10 steps.

**Step 0) Initialization**

The weights are randomly initialized to the range  $(-1, 1)$ .

**Forward pass****Step 1) Input layer**

$x_1 =$

$x_2 =$

**Step 2) Hidden layer**

$$a_1^{(2)} = \sigma(w_{11}^{(2)}x_1 + w_{21}^{(2)}x_2) =$$

$$a_2^{(2)} = \sigma(w_{12}^{(2)}x_1 + w_{22}^{(2)}x_2) =$$

$$a_3^{(2)} = \sigma(w_{13}^{(2)}x_1 + w_{23}^{(2)}x_2) =$$

**Step 3) Output layer**

$$a_1^{(3)} = \sigma(w_{11}^{(3)}a_1^{(2)} + w_{21}^{(3)}a_2^{(2)} + w_{31}^{(3)}a_3^{(2)}) =$$

**Step 4) Calculate the cost**

$$E = \frac{1}{2}(y - a_1^{(3)})^2 =$$





# Tinker With a Neural Network Right Here in Your Browser.

## Don't Worry, You Can't Break It. We Promise.



Epoch  
000,000

Learning rate  
0.03

Activation  
Tanh

Regularization  
None

Regularization rate  
0

Problem type  
Classification

### DATA

Which dataset do you want to use?



Ratio of training to test data: 50%

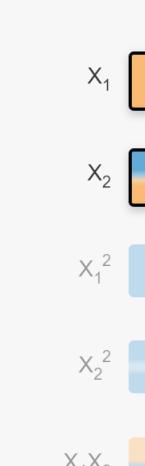
Noise: 0

Batch size: 10

REGENERATE

### FEATURES

Which properties do you want to feed in?



### 2 HIDDEN LAYERS

+

-

4 neurons



+

-

2 neurons



This is the output from one neuron. Hover to see it larger.

The outputs are mixed with varying weights, shown by the thickness of the lines.

### OUTPUT

Test loss 0.513  
Training loss 0.507

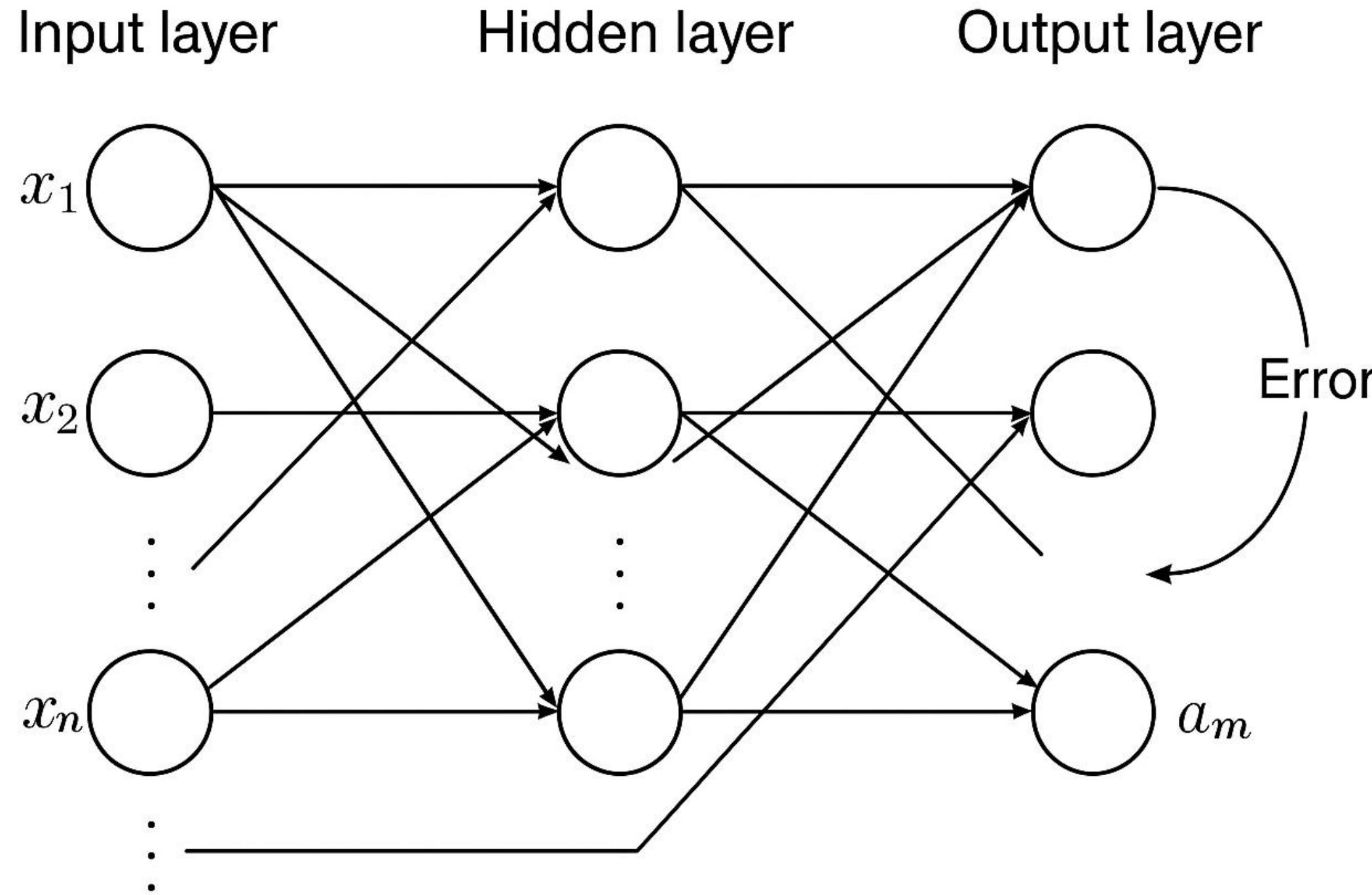


Colors shows data, neuron and weight values.  
-1 0 1

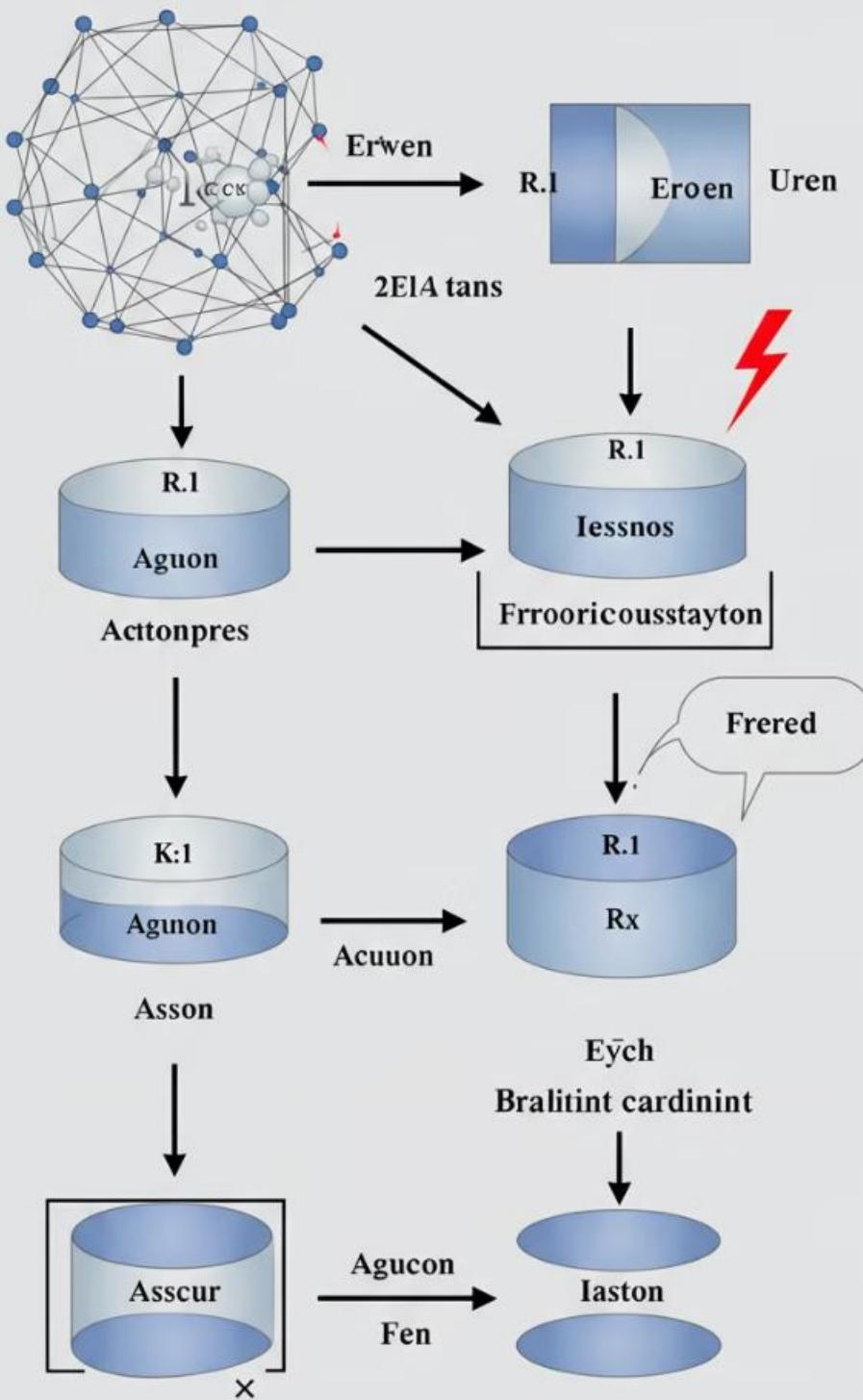
Show test data  Discretize output



# Backpropagation



Netunaet do frror ae protoves



# Proces učenia – spätná propagácia



## Inicializácia váh

Náhodné nastavenie počiatočných váh siete.

Poznámka: Správna inicializácia je kritická - príliš veľké alebo malé hodnoty môžu spomaliť alebo úplne zastaviť konvergenciu.

## Forward pass

Výpočet výstupu siete pre vstupné dátu.

Poznámka: Každý neurón aplikuje aktivačnú funkciu na vážený súčet svojich vstupov a postupuje signál ďalej.

## Výpočet chyby

Porovnanie výstupu so správnou hodnotou.

Poznámka: Používajú sa rôzne stratové funkcie ako MSE, cross-entropy alebo hinge loss, v závislosti od typu úlohy.

## Backward pass

Spätné šírenie chyby a aktualizácia váh:  $w = w - \eta \cdot \frac{\partial \text{Loss}}{\partial w}$

Poznámka: Parameter  $\eta$  (learning rate) určuje veľkosť kroku pri úprave váh. Príliš veľká hodnota môže spôsobiť nestabilitu, príliš malá pomalú konvergenciu.

# Keras



# Tensorflow

## Prehľad knižnice Keras

### </> High-level API

Postavené nad TensorFlow, s jednoduchou syntaxou. Umožňuje rýchle prototypovanie a zjednodušuje implementáciu komplexných modelov bez potreby písania nízkourovňového kódu.



### Vrstvy a modely

Preddefinované vrstvy a flexibilné modely. Poskytuje širokú škálu neurónových vrstiev (Dense, Conv2D, LSTM) a možnosť vytvárať vlastné architektúry pomocou funkčného a sekvenčného API.



### Nástroje

Optimalizátory, metriky a funkcie straty. Obsahuje rôzne optimalizačné algoritmy (Adam, SGD, RMSprop), funkcie straty (categorical\_crossentropy, MSE) a metriky (accuracy, precision) pre efektívne trénovanie modelov.



### GPU podpora

Efektívne využitie grafických procesorov. Automatická optimalizácia výpočtov na GPU hardvéri, čo výrazne zrýchluje proces trénovania neurónových sietí aj pri rozsiahlych dátových súboroch.

KERAS 3.0 RELEASED

# A superpower for ML developers

Keras is a deep learning API designed for human beings, not machines. Keras focuses on debugging speed, code elegance & conciseness, maintainability, and deployability. When you choose Keras, your codebase is smaller, more readable, easier to iterate on.

[API DOCS](#)[GUIDES](#)[EXAMPLES](#)

```
inputs = keras.Input(shape=(32, 32, 3))
x = layers.Conv2D(32, 3, activation="relu")(inputs)
x = layers.Conv2D(64, 3, activation="relu")(x)
residual = x = layers.MaxPooling2D(3)(x)

x = layers.Conv2D(64, 3, padding="same")(x)
x = layers.Activation("relu")(x)
x = layers.Conv2D(64, 3, padding="same")(x)
x = layers.Activation("relu")(x)
x = x + residual
```



## Welcome to multi-framework machine learning

With its multi-backend approach, Keras gives you the freedom to work with JAX, TensorFlow, and PyTorch.



Search projects



Help

Sponsors

Log in

Register

# keras 3.9.2

`pip install keras`

Latest version

Released: Apr 2, 2025

Multi-backend Keras

## Navigation

 [Project description](#) [Release history](#) [Download files](#)

## Verified details

*These details have been [verified by PyPI](#)*

## Maintainers



fchollet



hertschuh



jeffcarp



tf-nightly

## Unverified details

## Project description

### Keras 3: Deep Learning for Humans

Keras 3 is a multi-backend deep learning framework, with support for JAX, TensorFlow, PyTorch, and OpenVINO (for inference-only). Effortlessly build and train models for computer vision, natural language processing, audio processing, timeseries forecasting, recommender systems, etc.

- **Accelerated model development:** Ship deep learning solutions faster thanks to the high-level UX of Keras and the availability of easy-to-debug runtimes like PyTorch or JAX eager execution.
- **State-of-the-art performance:** By picking the backend that is the fastest for your model architecture (often JAX!), leverage speedups ranging from 20% to 350% compared to other frameworks. [Benchmark here](#).
- **Datacenter-scale training:** Scale confidently from your laptop to large clusters of GPUs or TPUs.

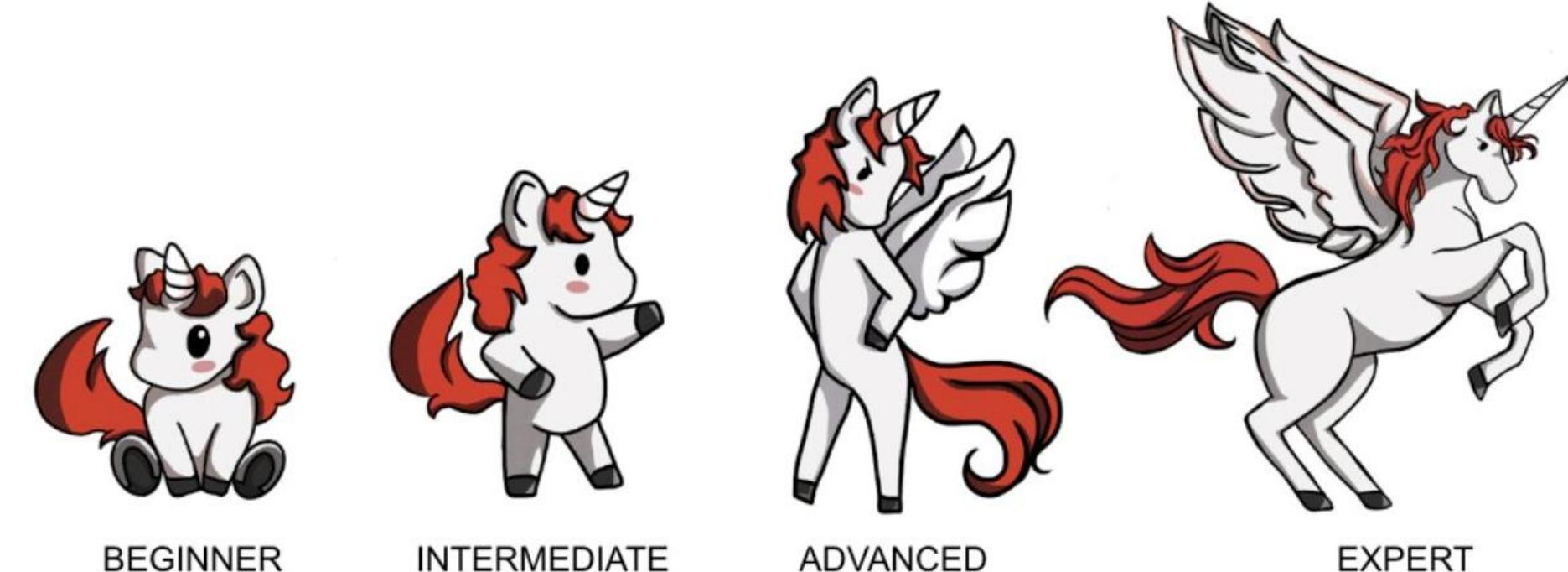
Join nearly three million developers, from burgeoning startups to global enterprises, in harnessing the power of Keras 3.

## Installation

### Install with pip

Keras 3 is available on PyPI as `keras`. Note that Keras 2 remains available as the `tf-keras` package.

# Inštalácia a verzia **Keras**



BEGINNER

INTERMEDIATE

ADVANCED

EXPERT

```
!pip install -q --upgrade keras-nlp
!pip install -q --upgrade keras # Upgrade to Keras 3.
```

```
import os

os.environ["KERAS_BACKEND"] = "jax" # or "tensorflow" or "torch"

import keras_nlp
import keras

# Use mixed precision to speed up all training in this guide.
keras.mixed_precision.set_global_policy("mixed_float16")
```

## Keras 3 API documentation

[Models API](#)[Layers API](#)[Callbacks API](#)[Ops API](#)[Optimizers](#)[Metrics](#)[Losses](#)[Data loading](#)

### Built-in small datasets

[MNIST digits classification dataset](#)[CIFAR10 small images classification dataset](#)[CIFAR100 small images classification dataset](#)

### IMDB movie review sentiment classification dataset

[Reuters newswire classification dataset](#)[Fashion MNIST dataset, an alternative to MNIST](#)[California Housing price regression dataset](#)[Keras Applications](#)[Mixed precision](#)[Multi-device distribution](#)

# IMDB movie review sentiment classification dataset

## load\_data function

[\[source\]](#)

```
keras.datasets.imdb.load_data(  
    path="imdb.npz",  
    num_words=None,  
    skip_top=0,  
    maxlen=None,  
    seed=113,  
    start_char=1,  
    oov_char=2,  
    index_from=3,  
    **kwargs  
)
```

Loads the [IMDB dataset](#).

This is a dataset of 25,000 movies reviews from IMDB, labeled by sentiment (positive/negative). Reviews have been preprocessed, and each review is encoded as a list of word indexes (integers). For convenience, words are indexed by overall frequency in the dataset, so that for instance the integer "3" encodes the 3rd most frequent word in the data. This allows for quick filtering operations such as: "only consider the top 10,000 most common words, but eliminate the top 20 most common words".

As a convention, "0" does not stand for a specific word, but instead is used to encode the pad token.

## Arguments

- **path**: where to cache the data (relative to `~/keras/dataset`).
- **num\_words**: integer or None. Words are ranked by how often they occur (in the training set) and only the `num_words` most frequent words are kept. Any less frequent word will appear as `oov_char` value in the sequence data. If None, all words are kept. Defaults to `None`.
- **skip\_top**: skip the top N most frequently occurring words (which may not be informative). These words will appear as `oov_char` value in the dataset. When 0, no words are skipped. Defaults to `0`.
- **maxlen**: int or None. Maximum sequence length. Any longer sequence will be truncated. None, means no truncation. Defaults to `None`.
- **seed**: int. Seed for reproducible data shuffling.
- **start\_char**: int. The start of a sequence will be marked with this character. 0 is usually the padding character. Defaults to `1`.
- **oov\_char**: int. The out-of-vocabulary character. Words that were cut out because of the `num_words` or `skip_top` limits will be replaced with this character.

IMDB movie review sentiment classification dataset

[load\\_data function](#)  
[get\\_word\\_index function](#)

# Large Movie Review Dataset

This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. We provide a set of 25,000 highly polar movie reviews for training, and 25,000 for testing. There is additional unlabeled data for use as well. Raw text and already processed bag of words formats are provided. See the README file contained in the release for more details.

## [Large Movie Review Dataset v1.0](#)

When using this dataset, please cite our ACL 2011 paper [[bib](#)].

## Contact

For comments or questions on the dataset please contact [Andrew Maas](#). As you publish papers using the dataset please notify us so we can post a link on this page.

## Publications Using the Dataset

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). [Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics \(ACL 2011\)](#).

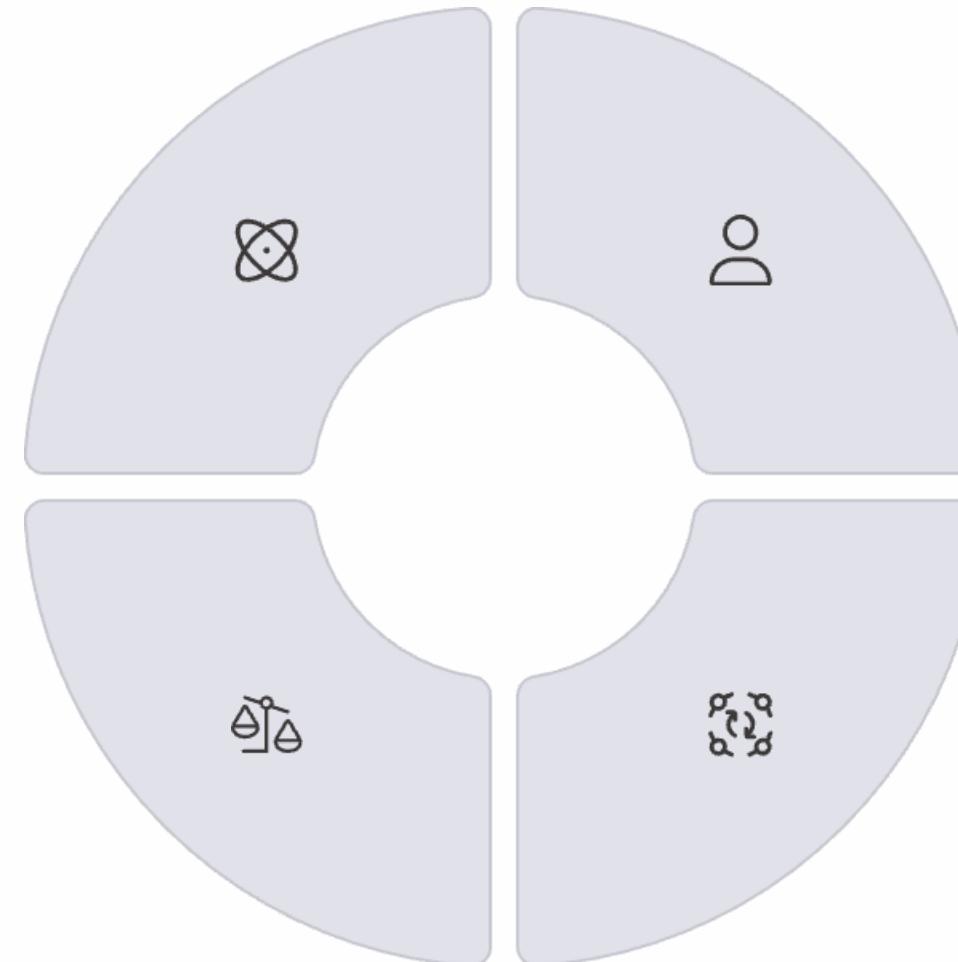
# Prečo Keras?

## Jednoduchosť

Jednoduché učenie a vývoj  
neurónových sietí.

## Škálovateľnosť

Ideálne pre začiatočníkov aj  
pokročilých.



## Kompatibilita

Plná integrácia s TensorFlow 2.x.

## Komunita

Veľká komunita a množstvo  
príkladov.

# Nastavenie prostredia pre Keras

## Inštalácia Python

Odporúčaný Python 3.9 alebo novší.

Stiahnite z [python.org](https://python.org) a postupujte podľa inštalačných pokynov pre váš operačný systém. Nezabudnite pridať Python do PATH.

## Inštalácia TensorFlow

`pip install tensorflow`

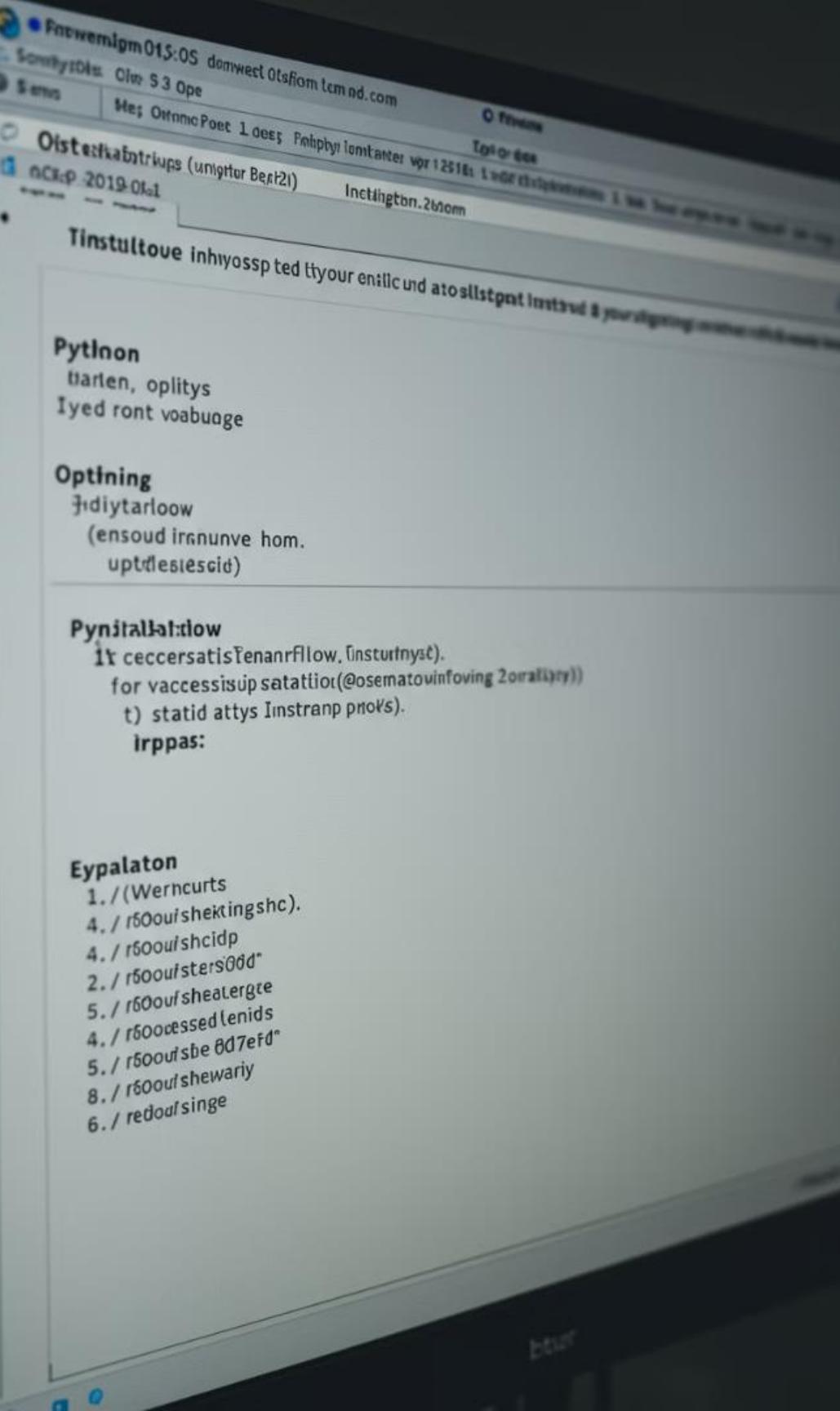
Tento príkaz môžete spustiť v príkazovom riadku alebo v terminále.

Overte si, že máte najnovšiu verziu pip nainštalovanú.

## Overenie inštalácie

`import tensorflow as tf; print(tf.__version__)`

Spusťte tento príkaz v Python konzole alebo v skripte. Malí by ste vidieť vypísanú verziu TensorFlow bez chybových hlásení.



# Kontrola GPU podpory



## Importovať TensorFlow

```
import tensorflow as tf
```

Poznámka: Uistite sa, že máte nainštalovanú správnu verziu TensorFlow kompatibilnú s vašou CUDA a cuDNN konfiguráciou.



## Kontrola GPU

```
tf.config.list_physical_devices('GPU')
```

Poznámka: Tento príkaz zobrazí zoznam všetkých GPU zariadení dostupných pre TensorFlow. Ak máte viacero GPU, všetky budú uvedené v zozname.



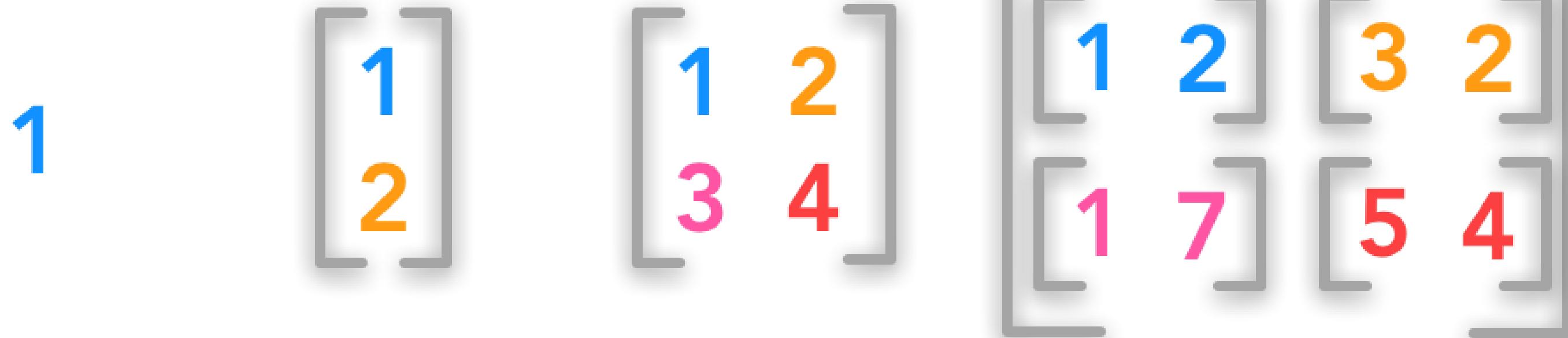
## Overenie výsledku

Neprázdný zoznam znamená dostupné GPU

Poznámka: Ak je výsledok prázdný zoznam [], vaše GPU buď nie je kompatibilné s TensorFlow alebo nie sú správne nainštalované ovládače. Skontrolujte inštaláciu CUDA a cuDNN.

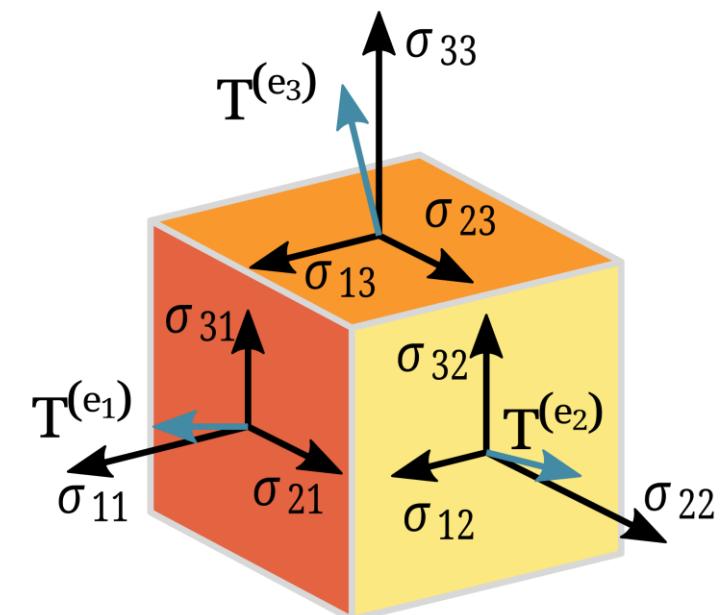
Poznámka: Pre optimálny výkon neurónovej siete je dôležité správne nastaviť GPU podporu. Trénovanie na GPU môže byť 10-50x rýchlejšie ako na CPU v závislosti od architektúry.

# Scalar   Vector   Matrix   Tensor



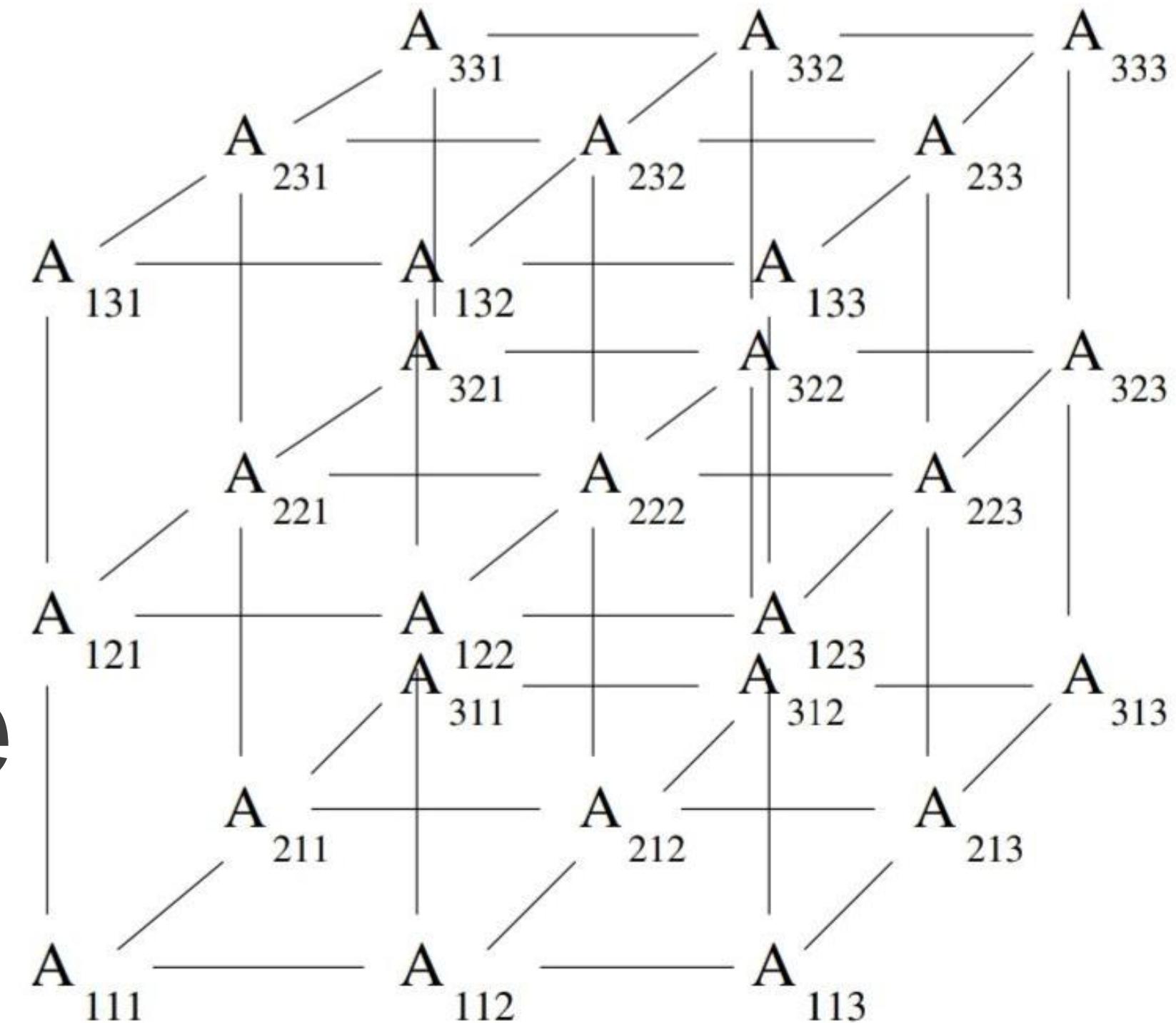
**Použitie v strojovom učení (napr. TensorFlow, PyTorch):**

1. Reprezentácia obrázov (napr.  $28 \times 28 \times 1$ )
2. Reprezentácia textovej sekvencie (napr.  $100 \times 50$ )
3. Vstup/výstup do neurónovej siete



# Vizualizácia Tenzora

## 3. úrovne v 3D priestore



# Fyzikálne Tenzory

- **Tenzor napäťia (Cauchyho tenzor):**

$$\sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{bmatrix}$$

→ popisuje vnútorné mechanické napäťia v pevnom telesе

- **Moment zotrvačnosti (tenzor zotrvačnosti):**

$$I = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

→ popisuje, ako sa telo otáča okolo osí

- **Dielektrická permitivita (v anizotropných materiáloch):**

$$\varepsilon = \begin{bmatrix} \varepsilon_{xx} & \varepsilon_{xy} & \varepsilon_{xz} \\ \varepsilon_{yx} & \varepsilon_{yy} & \varepsilon_{yz} \\ \varepsilon_{zx} & \varepsilon_{zy} & \varepsilon_{zz} \end{bmatrix}$$

→ popisuje, ako materiál reaguje na elektrické pole

- **Riemannov tenzor zakrivenia (4. rádu):**

$$R^i_{jkl}$$

→ používa sa v Einsteinovej všeobecnej teórii relativity, časopriestoru

# Nastavenie náhodného seedu

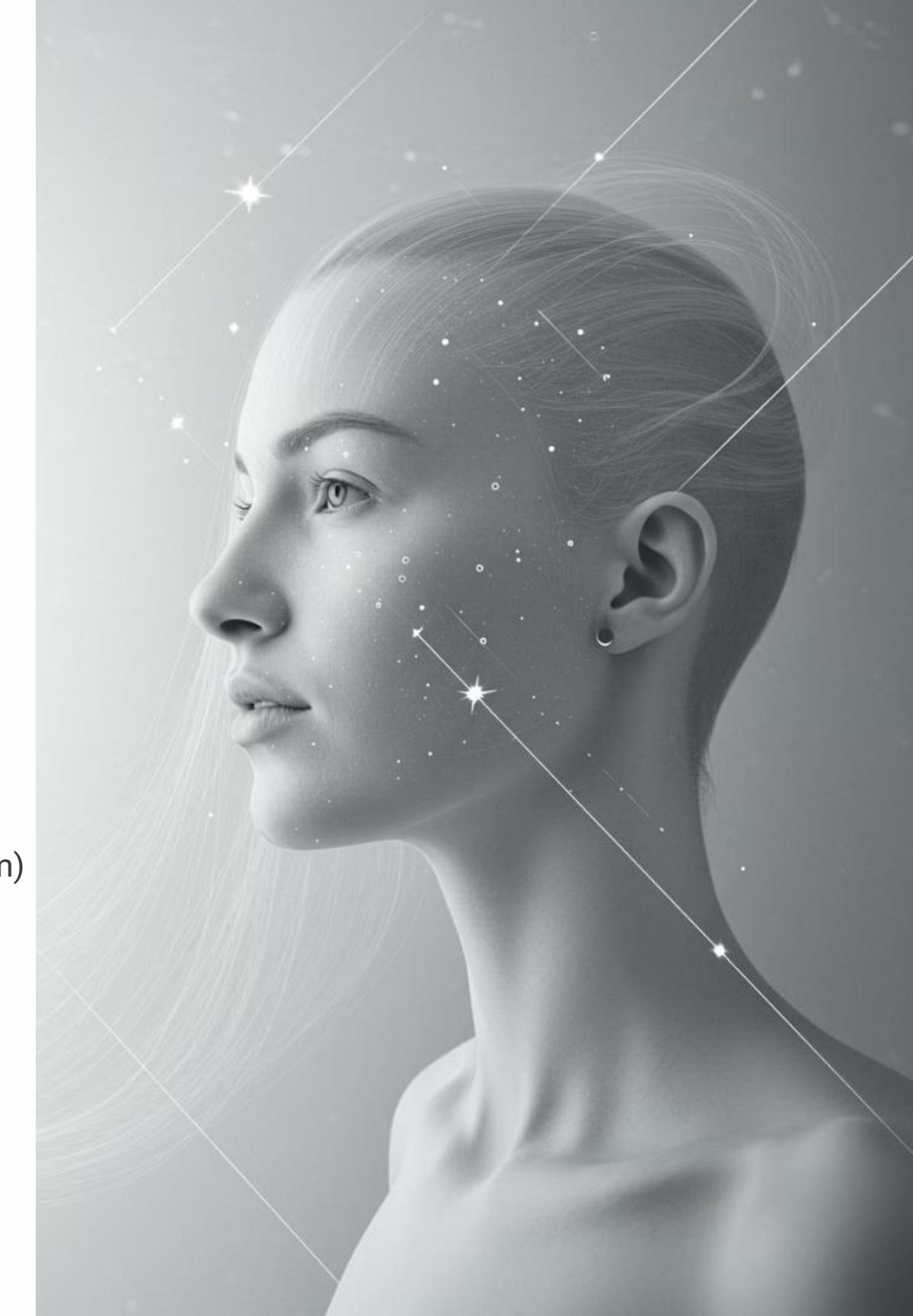
```
import numpy as np  
import tensorflow as tf  
  
np.random.seed(42)  
tf.random.set_seed(42)
```

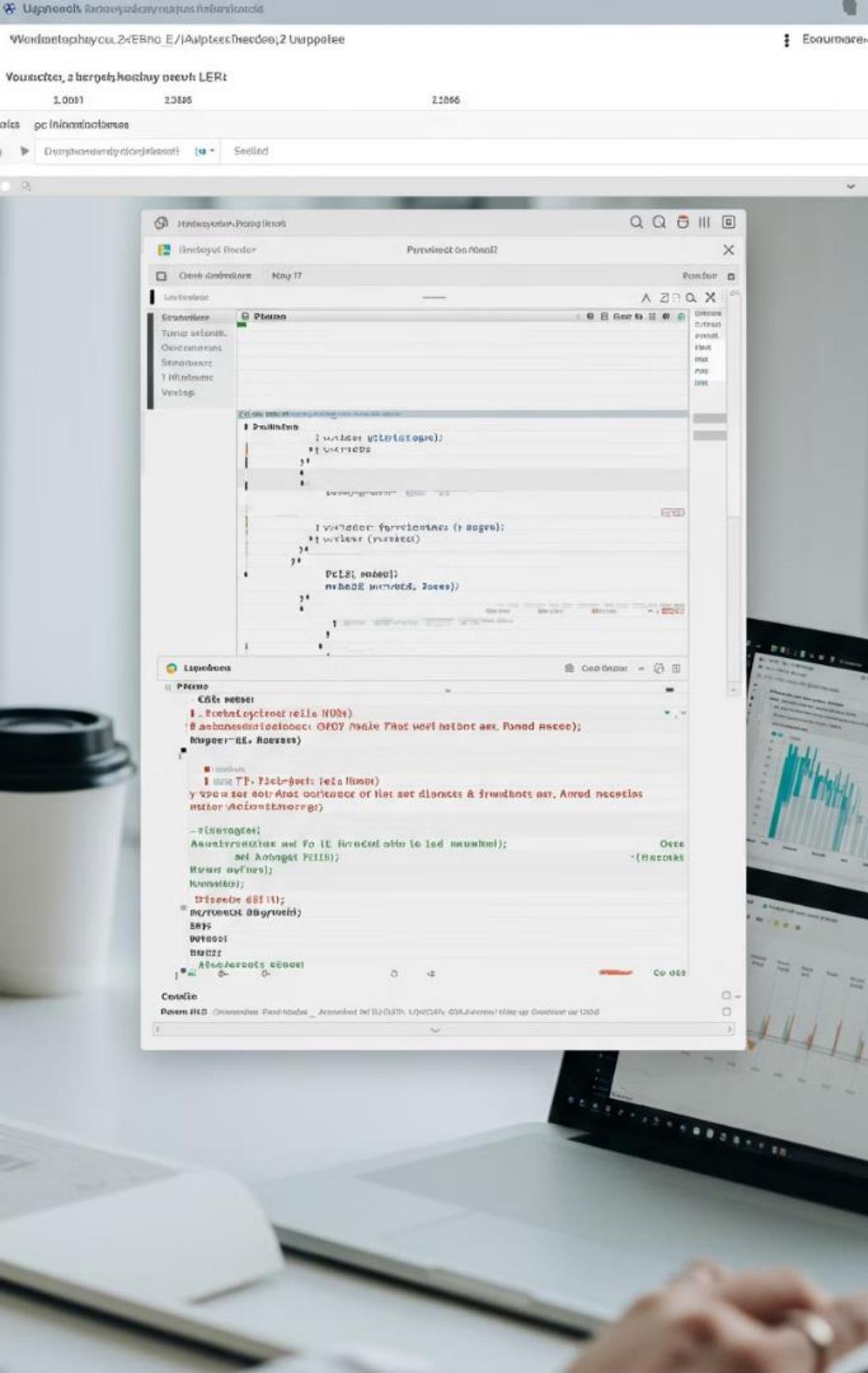
Nastavenie náhodného seedu zabezpečí replikovateľnosť experimentov. Rovnaký seed znamená rovnaké náhodné inicializácie váh.

Dôležité poznámky:

- Seed hodnota (napr. 42) môže byť ľubovoľné celé číslo
- Musíte nastaviť seed pre všetky používané knižnice (NumPy, TensorFlow, Python random)
- Nastavenie seedu ovplyvňuje inicializáciu váh, výber trénovacích vzoriek a všetky náhodné operácie
- Pre produkčné nasadenie zväžte vypnutie fixného seedu
- Nastavenie seedu nefunguje spoľahlivo v distribuovanom trénovaní

Pri debugovaní a experimentovaní je fixný seed nevyhnutný pre konzistentné výsledky medzi spusteniami.





# Jupyter Notebook pre Keras



## Inštalácia Jupyter

pip install notebook

Jupyter Notebook je webové rozhranie pre interaktívne programovanie. Inštalácia vyžaduje Python 3.x a správcu balíkov pip.



## Spustenie servera

jupyter notebook

Tento príkaz spustí lokálny server na porte 8888. V prehliadači sa automaticky otvorí domovská stránka s adresárovou štruktúrou.



## Vytvorenie nového súboru

Kliknite na "New" a vyberte "Python 3"

Vytvorí sa nový notebook s príponou .ipynb. Bunky môžete spúštať pomocou Shift+Enter. Striedajte kódové bunky a textové bunky pre prehľadnú dokumentáciu.

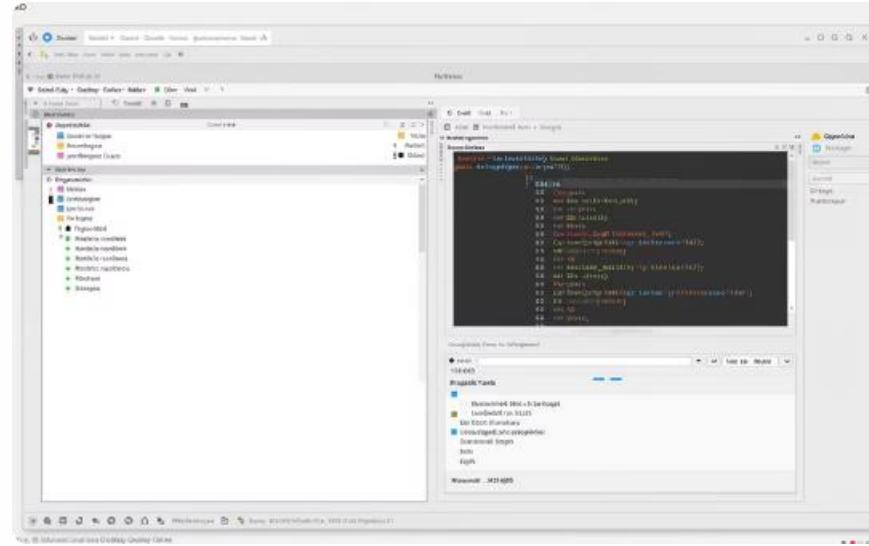


## Test importu Keras

from tensorflow import keras

Keras je vysokoúrovňové API pre prácu s neurónovými sieťami. Spustením tohto príkazu overíte, či máte TensorFlow a Keras správne nainštalované a pripravené na použitie.

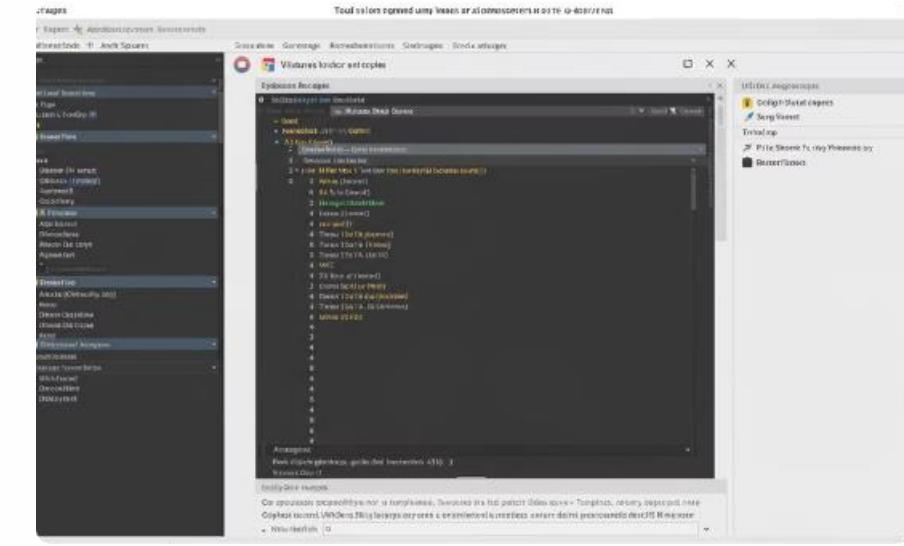
# Odporúčané vývojové prostredia



## Google Colab

Cloudové riešenie s bezplatným prístupom k GPU. Nevyžaduje lokálnu inštaláciu.

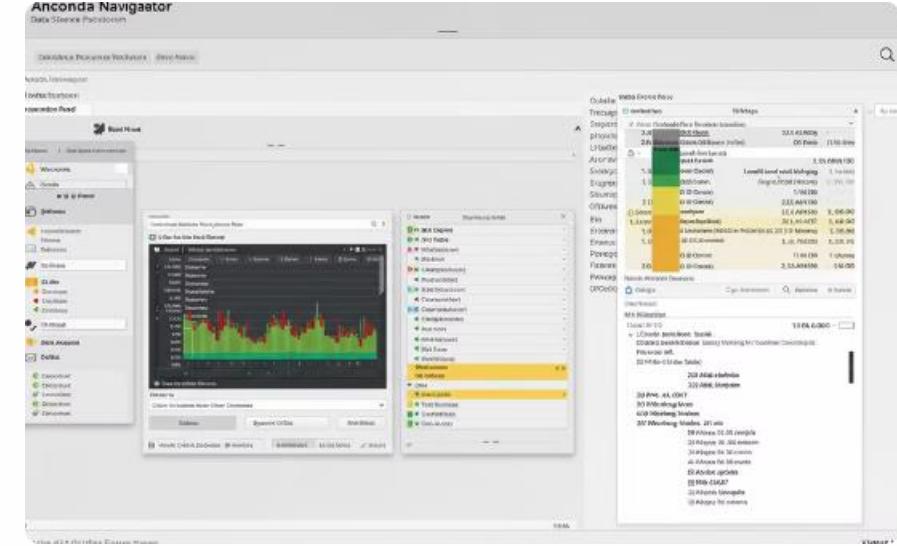
Ideálne pre začiatočníkov, umožňuje zdieľanie notebookov a spoluprácu v reálnom čase. Podporuje integráciu s Google Drive pre jednoduché ukladanie projektov.



## VS Code

Lokálny editor s výbornou podporou pre Python a Jupyter notebooky.

Ponúka rozšírenia pre Keras a TensorFlow s inteligentným dokončovaním kódu. Umožňuje ladenie kódu priamo v editore a vizualizáciu výsledkov trénovania.



## Anaconda

Distribúcia Pythonu so zabudovanými nástrojmi pre dátovú vedu.

Obsahuje Spyder IDE, Jupyter a množstvo predstalovaných balíčkov pre strojové učenie. Správca balíčkov Conda uľahčuje inštaláciu a správu všetkých potrebných knižníc bez konfliktov.



Príkazy

+ Kód

+ Text

Kopírovať na Disk

✓ T4 Operačná pamäť  
Disk

## Obsah



Image classification via fine-tuning with EfficientNet



Introduction: what is EfficientNet



### Setup and data loading



Loading data



Visualizing the data



Data augmentation



Prepare inputs

Training a model from scratch

Transfer learning from pre-trained weights

Tips for fine tuning EfficientNet

## + Sekcia

## Image classification via fine-tuning with EfficientNet

**Author:** [Yixing Fu](#)

**Date created:** 2020/06/30

**Last modified:** 2023/07/10

**Description:** Use EfficientNet with weights pre-trained on imagenet for Stanford Dogs classification.

### Introduction: what is EfficientNet

EfficientNet, first introduced in [Tan and Le, 2019](#) is among the most efficient models (i.e. requiring least FLOPS for inference) that reaches State-of-the-Art accuracy on both imagenet and common image classification transfer learning tasks.

The smallest base model is similar to [MnasNet](#), which reached near-SOTA with a significantly smaller model. By introducing a heuristic way to scale the model, EfficientNet provides a family of models (B0 to B7) that represents a good combination of efficiency and accuracy on a variety of scales. Such a scaling heuristics (compound-scaling, details see [Tan and Le, 2019](#)) allows the efficiency-oriented base model (B0) to surpass models at every scale, while avoiding extensive grid-search of hyperparameters.

A summary of the latest updates on the model is available at [here](#), where various augmentation schemes and semi-supervised learning approaches are applied to further improve the imagenet performance of the models. These extensions of the model can be used by updating weights without changing model architecture.

### B0 to B7 variants of EfficientNet

*(This section provides some details on "compound scaling", and can be skipped if you're only interested in using the models)*

Based on the [original paper](#) people may have the impression that EfficientNet is a continuous family of models created by arbitrarily choosing scaling factor in as Eq.(3) of the paper. However, choice of resolution, depth and width are also restricted by many factors:

# NPZ Numpy's compressed array format

Extensions: `.npz`

NPZ is a file format by numpy that provides storage of array data using gzip compression. This imageio plugin supports data of any shape, and also supports multiple images per file.

However, the npz format does not provide streaming; all data is read/written at once. Further, there is no support for meta data.

Beware that the numpy npz format has a bug on a certain combination of Python 2.7 and numpy, which can cause the resulting files to become unreadable on Python 3. Also, this format is not available on Pypy.

See the BSDF format for a similar (but more fully featured) format.

## Parameters for reading

None

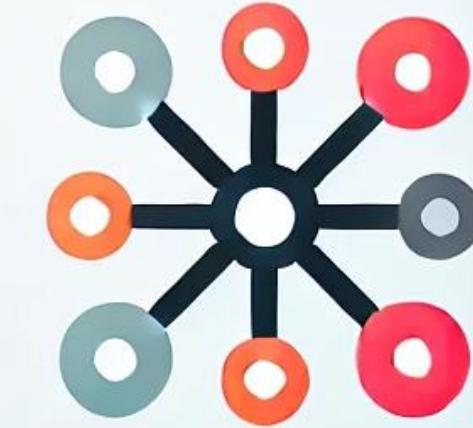
## Parameters for saving

None

# Úlohy

# Neur. Siete

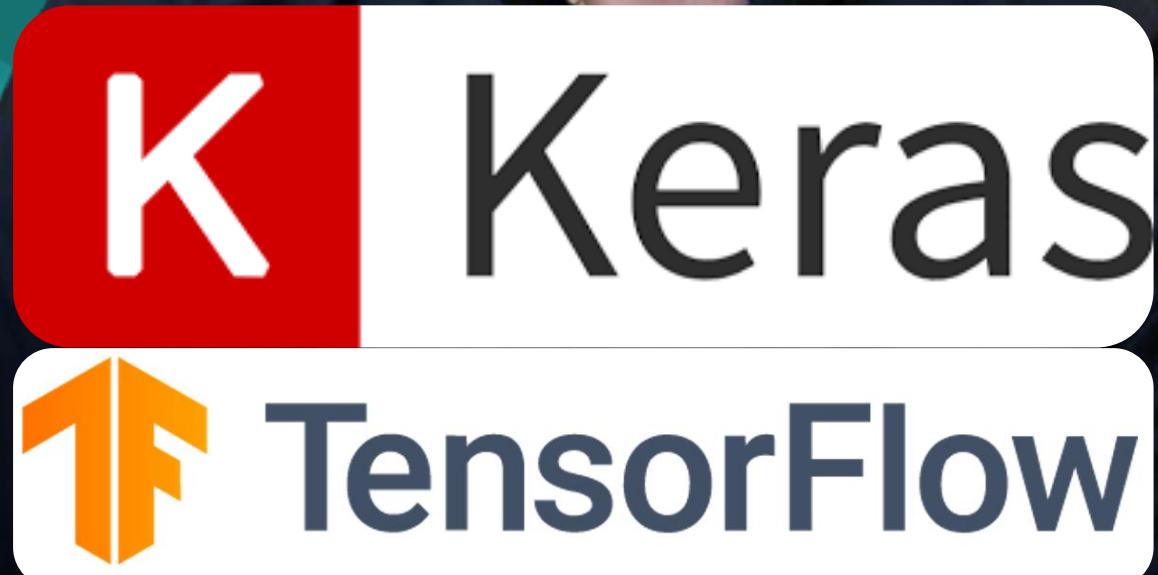
1. Čo sú to neurónové siete?
2. Z čoho sa skladá umelý neurón?
3. Aké sú najčastejšie typy aktivačných funkcií?
4. Čo je to Keras?
5. Na čo slúži metóda compile()?
6. Aký je rozdiel medzi Sequential a Functional API v Keras?



Keras®

# Ako na Vytvorenie Neur. Siete

AKREDITOVANÝ KURZ





# Čo sa naučíme?

1. Čo je to sekvenčný model?
2. Ako sa pridávajú vrstvy do modelu?
3. Na čo slúži komplilácia modelu?
4. Ako prebieha trénovanie modelu?
5. Ako sa vyhodnocuje výkonnosť modelu?
6. Čo je to predikcia a ako sa vykonáva?
7. Profit

# Základná štruktúra modelu – Sequential

## Definícia

Sequential model predstavuje lineárne usporiadanú neurónovú sieť, kde vrstvy nasledujú jedna za druhou.

```
from tensorflow.keras.models import Sequential  
model = Sequential()
```

*Poznámka: Tento spôsob vytvárania modelu je vhodný, keď chceme pridávať vrstvy postupne pomocou metódy add().*

*Používa sa najmä pri experimentovaní a postupnom budovaní siete.*

*Poznámka: Sequential model je najjednoduchší typ modelu v Keras. Pre komplexnejšie architektúry s vetvením alebo viacerými vstupmi/výstupmi je potrebné použiť funkcionálne API alebo subclassing.*

## Alternatívny zápis

Vrstvy môžeme definovať aj priamo pri vytvorení modelu ako zoznam.

```
model = Sequential([  
    Dense(128, activation='relu',  
          input_shape=(784,)),  
    Dense(10, activation='softmax')  
])
```

*Poznámka: Tento kompaktnejší zápis je užitočný, keď máme jasné predstavu o architektúre modelu. Parameter input\_shape=(784,) definuje rozmer vstupných dát a čísla 128 a 10 predstavujú počet neurónov v jednotlivých vrstvách.*

# Pridávanie vrstiev – Dense

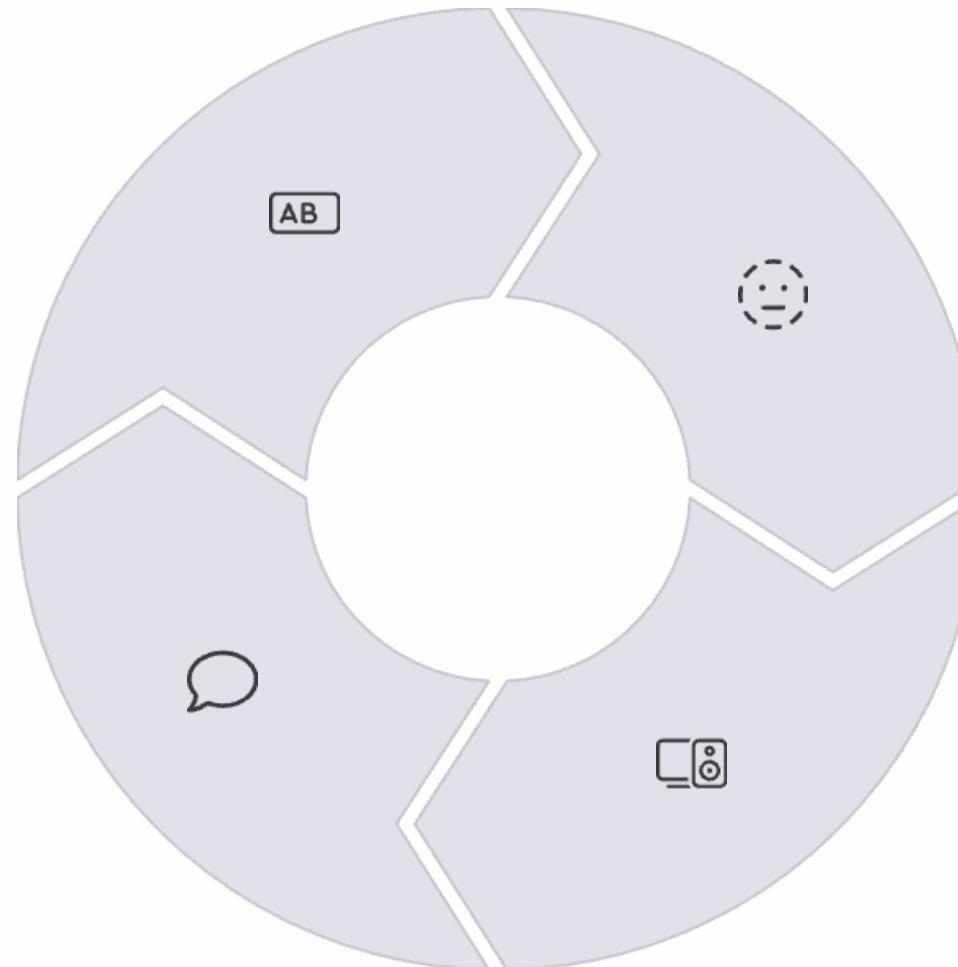
## Vstupná vrstva

Definuje tvar vstupných dát pomocou `input_shape`. Určuje počet vstupných parametrov, ktoré model očakáva.

Napr. `input_shape=(784,)` pre obrázky MNIST.

## Aktivačné funkcie

Pridávajú nelinearitu do modelu. Populárne aktivačné funkcie zahŕňajú ReLU pre skryté vrstvy, sigmoid pre binárnu klasifikáciu a softmax pre viac triednu klasifikáciu. Bez aktivačných funkcií by sa model správal ako lineárna regresia.



## Skryté vrstvy

Spracúvajú dátu pomocou váh a aktivačných funkcií. Počet neurónov v skrytých vrstvách ovplyvňuje kapacitu modelu učiť sa. Viac neurónov umožňuje zachytíť komplexnejšie vzory v dátach.

## Výstupná vrstva

Produkuje finálny výsledok siete. Počet neurónov závisí od typu úlohy - jeden neurón pre binárnu klasifikáciu, viac neurónov pre viac triednu klasifikáciu (počet tried) alebo regresiu.

# Príklad modelu (Sequential API)

```
from tensorflow.keras.models import Sequential # Import sekvenčného modelu
from tensorflow.keras.layers import Dense      # Import dense vrstvy

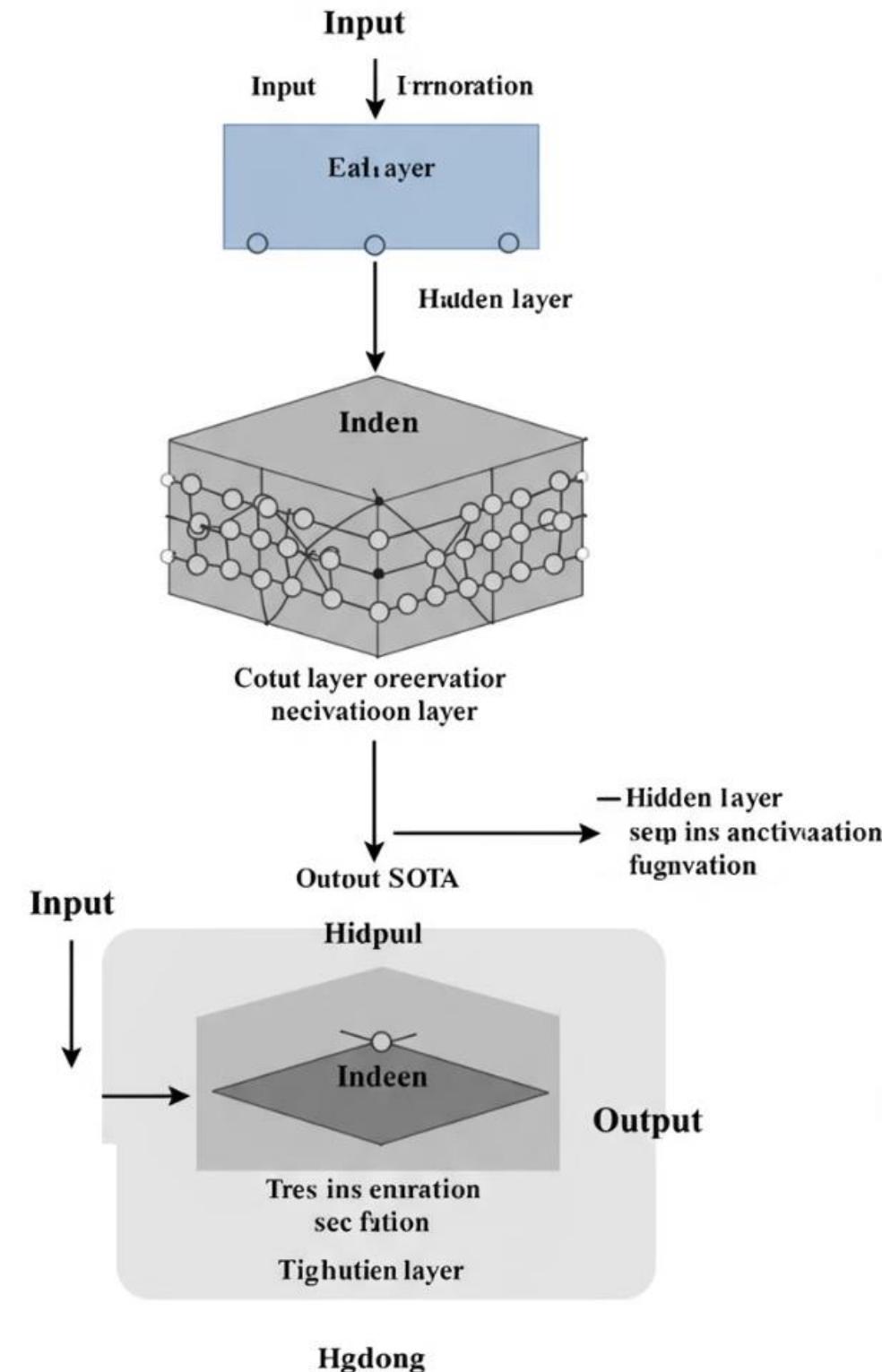
# Vytvorenie sekvenčného modelu s dvoma vrstvami
model = Sequential([
    # Prvá vrstva - vstupná + skrytá:
    #   - 32 neurónov
    #   - ReLU aktivačná funkcia
    #   - 10-dimenzionálny vstupný vektor
    Dense(32, activation='relu', input_shape=(10,)),

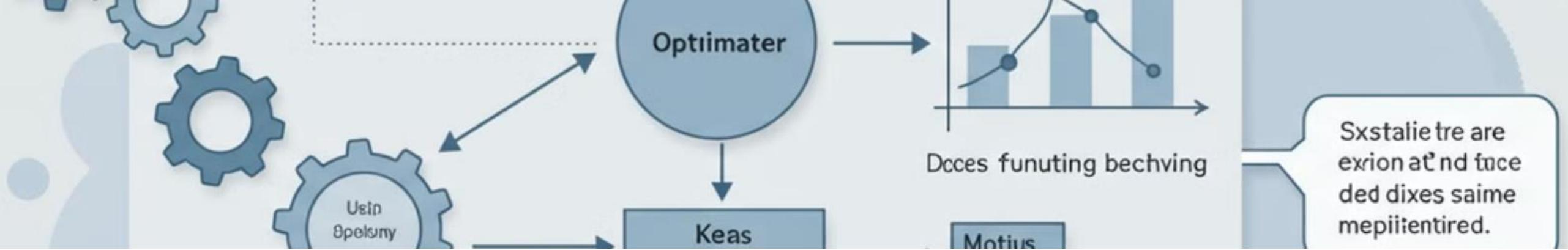
    # Výstupná vrstva:
    #   - 1 neurón pre binárnu klasifikáciu
    #   - Sigmoid aktívacia (výstup medzi 0 a 1)
    Dense(1, activation='sigmoid')
])
```

Tento jednoduchý model obsahuje vstupnú vrstvu s 10 vstupmi, skrytú vrstvu s 32 neurónmi a výstupnú vrstvu s 1 neurónom pre binárnu klasifikáciu.

Poznámky k štruktúre:

- **Sequential API** - umožňuje jednoduché spájanie vrstiev v lineárnej postupnosti
- **Dense vrstva** - plne prepojená vrstva, kde každý neurón prijíma vstupy zo všetkých neurónov predchádzajúcej vrstvy
- **ReLU aktívacia** - štandardná voľba pre skryté vrstvy, zavádza nelinearitu
- **Sigmoid aktívacia** - vhodná pre binárnu klasifikáciu, produkuje pravdepodobnosť medzi 0 a 1





# Kompilácia modelu

Kompilácia je kľúčovým krokom pred tréningom, kde definujeme základné prvky učenia.



## Optimalizátor

Algoritmus pre aktualizáciu váh (adam, sgd, rmsprop). Určuje ako model aktualizuje svoje parametre na základe gradientu. Adam je populárna voľba pre väčšinu úloh vďaka adaptívnej rýchlosťi učenia.



## Funkcia straty

Miera chyby modelu (binary\_crossentropy, mse). Definuje, ako model vyhodnocuje svoj výkon - čím nižšia hodnota, tým lepší model. Výber závisí od typu úlohy: klasifikácia vs. regresia.



## Metriky

Meranie výkonu modelu (accuracy, mae). Na rozdiel od funkcie straty, metriky slúžia len na monitorovanie, nie na optimalizáciu. Accuracy (presnosť) je štandardná metrika pre klasifikačné úlohy.

```
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

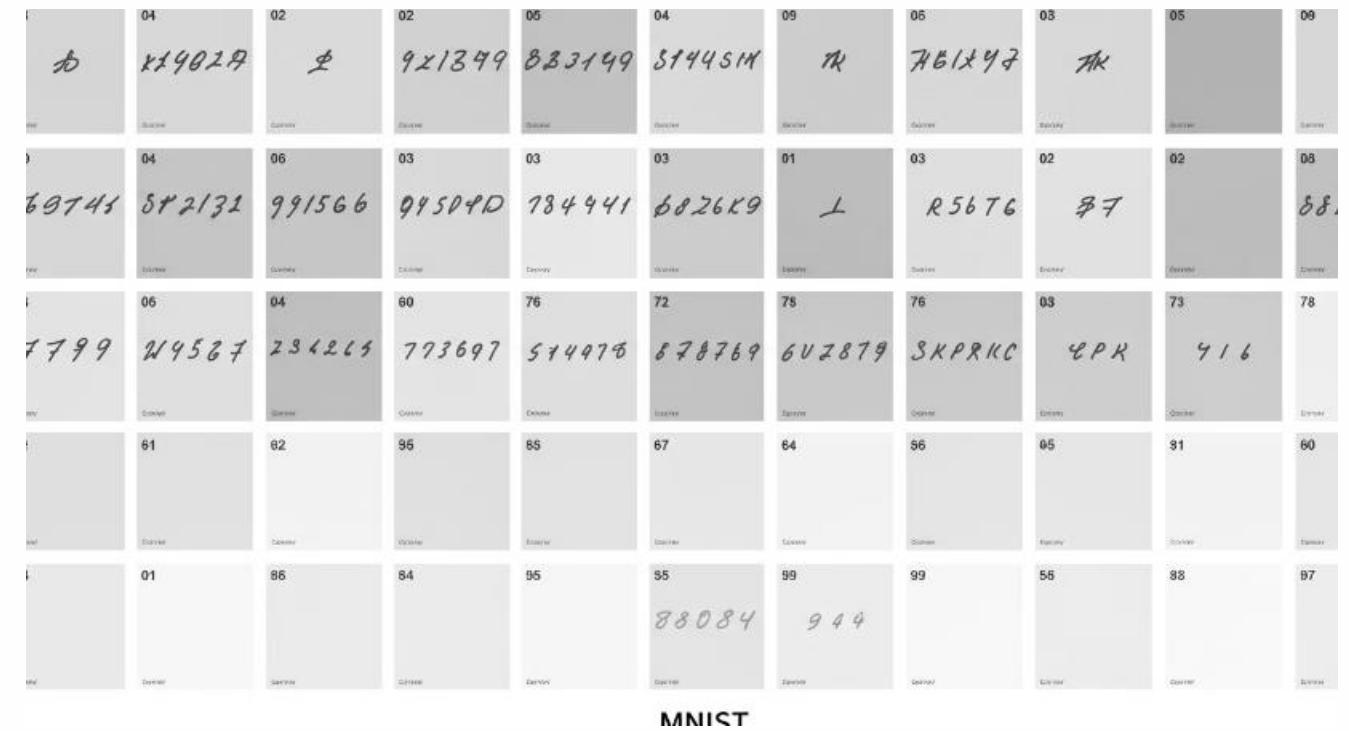
Príklad komplikácie modelu pre klasifikáciu do viacerých tried. Sparse\_categorical\_crossentropy je vhodná, keď sú cieľové triedy reprezentované ako celé čísla, nie one-hot vektory. Adam optimalizátor automaticky upravuje rýchlosť učenia počas tréningu.

# Dataset MNIST

## Charakteristika

Dataset ručne písaných číslíc používaný na testovanie algoritmov strojového učenia.

- 60 000 trénovacích obrázkov
- 10 000 testovacích obrázkov
- Rozmery: 28x28 pixelov (784 vstupov)
- Šedotónové obrázky (0-255)



Ukážka obrázkov z datasetu MNIST - ručne písané číslice od 0 do 9.

# Načítanie a predspracovanie dát MNIST

## Načítanie datasetu



```
from tensorflow.keras.datasets import mnist  
  
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

Pomocou knižnice Keras jednoducho importujeme dataset MNIST. Funkcia `load_data()` automaticky stiahne dáta ak nie sú lokálne dostupné a rozdelí ich na trénovaciu a testovaciu množinu.



## Zmena tvaru dát

```
X_train = X_train.reshape(-1, 28*28)
```

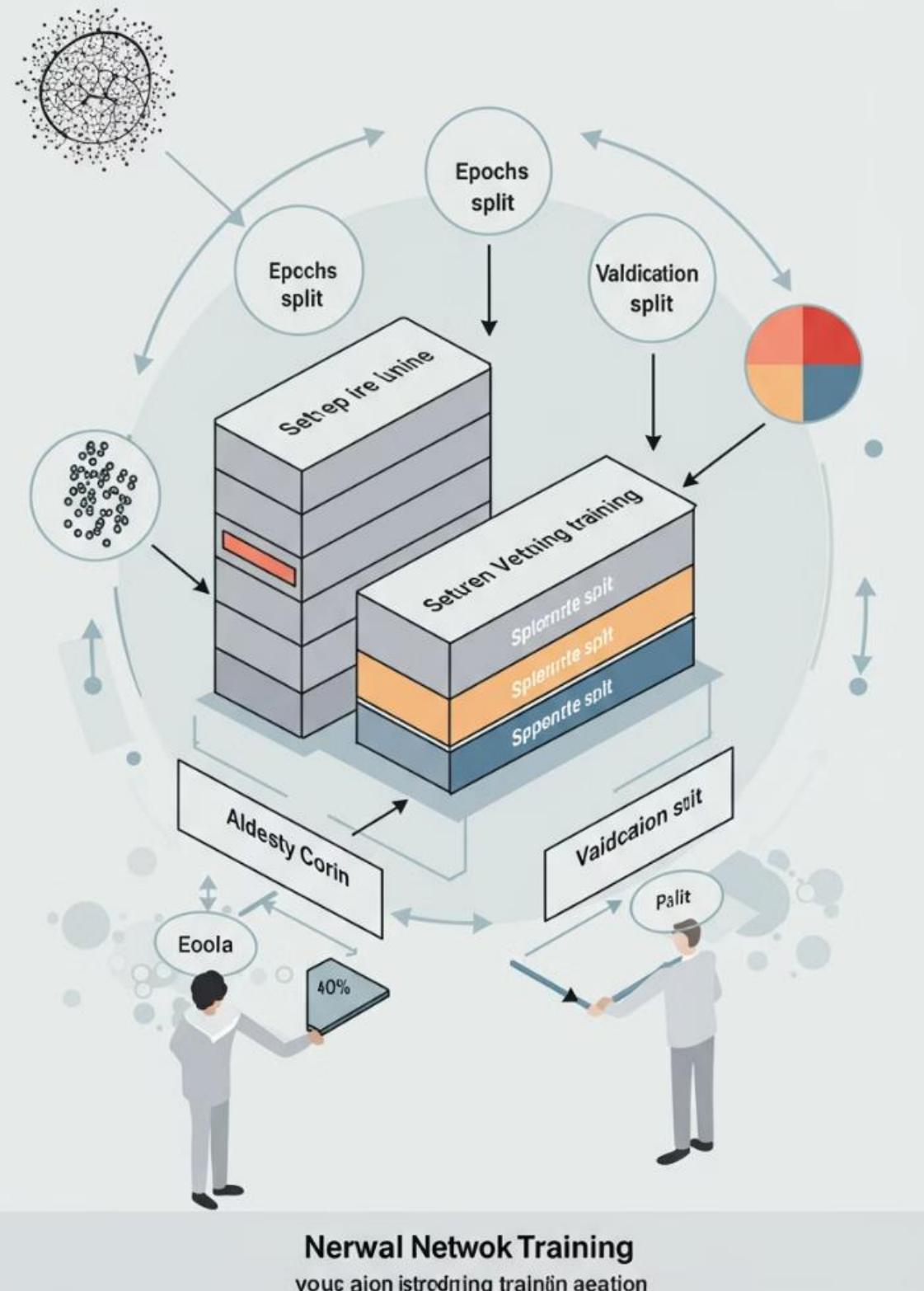
Transformujeme 2D obrázky (28x28 pixelov) na 1D vektory s dĺžkou 784 prvkov. Parameter -1 automaticky vypočíta počet vzoriek, čo umožňuje modelu spracovať dáta v požadovanom formáte.

3

## Normalizácia hodnôt

```
X_train = X_train.astype('float32') / 255
```

Konvertujeme celé čísla (0-255) na desatinné čísla v rozsahu 0-1. Táto normalizácia zlepšuje konvergenciu modelu, urýchľuje tréning a zvyšuje presnosť predikcie.



# Tréning modelu – fit()

Funkcia `fit()` je klúčová pre trénovanie neurónových sietí. Inicializuje proces učenia a optimalizácie váh siete.

## Parametre tréningu

- `epochs` - počet prejdení celým datasetom
- `batch_size` - veľkosť dávky dát
- `validation_split` - časť dát na validáciu

Správne nastavenie parametrov je klúčové pre efektívny tréning. Príliš mnoho epoch môže viesť k pretrénovaniu, zatiaľ čo príliš malé dávky môžu spomaliť učenie.

## Príklad použitia

```
history = model.fit(  
    X_train, y_train,  
    epochs=5,  
    batch_size=32,  
    validation_split=0.2  
)
```

Tento kód trénuje model na trénovacích dátach počas 5 epoch, s dávkami veľkosti 32 a vyčlení 20% dát na priebežnú validáciu.

## Výstup tréningu

Objekt `history` obsahuje metriky z priebehu tréningu (presnosť, strata).

Tieto metriky môžete vizualizovať pomocou `matplotlib` pre analýzu učenia modelu. Dôležité je sledovať najmä rozdiel medzi tréningovými a validačnými metrikami pre zistenie pretrénovania.

# Vyhodnotenie modelu – evaluate()

## Princíp

Metóda evaluate() vypočíta stratu a metriky na testovacích dátach. Je kľúčová pre overenie generalizačnej schopnosti modelu.

Poznámka: Testovacia sada musí zostať oddelená od tréningových dát, aby sme dosiahli objektívne hodnotenie výkonu modelu v reálnych podmienkach.

Poznámka: Vyhodnotenie modelu by malo nasledovať po tréningu modelu a malo by byť vykonané na dátach, ktoré model ešte nevidel. Nízka presnosť na testovacích dátach v porovnaní s tréningovými môže indikovať overfitting.

## Použitie

```
loss, accuracy = model.evaluate(X_test, y_test)  
print(f"Presnosť: {accuracy:.2f}")
```

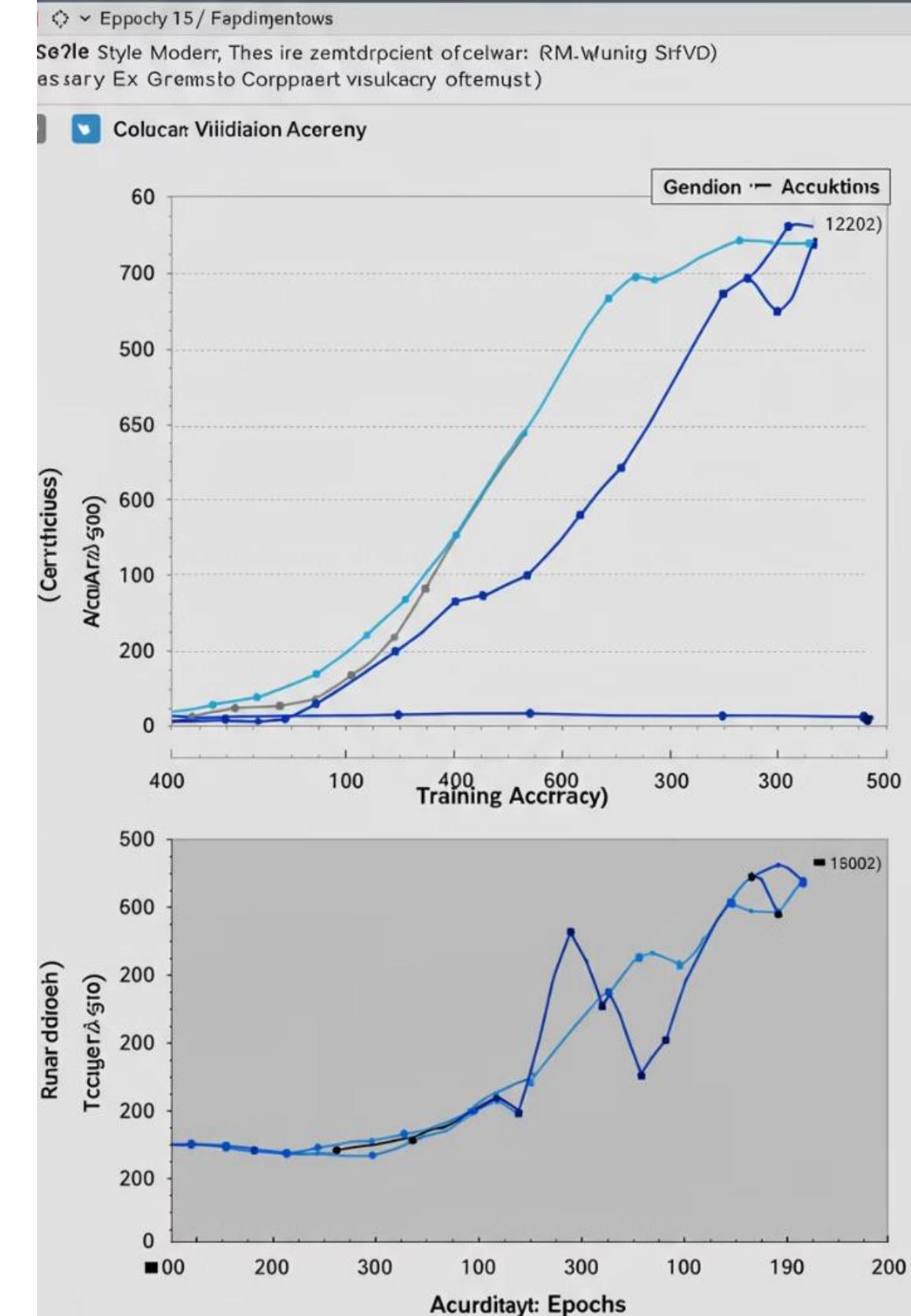
Poznámka: Môžeme použiť parameter verbose=0 pre potlačenie výpisu procesu vyhodnocovania a batch\_size pre úpravu veľkosti dávky pri vyhodnocovaní.

# Vizualizácia učenia

```
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Presnosť počas učenia')
plt.xlabel('Epochy')
plt.ylabel('Presnosť')
plt.legend(['Tréning', 'Validácia'])
plt.show()
```

Vizualizácia priebehu učenia pomáha identifikovať problémy ako overfitting (ked' tréningová presnosť rastie, ale validačná stagnuje alebo klesá).



# Úlohy Tvorba Sietí

1. Čo je to sekvenčný model?
2. Ako sa pridávajú vrstvy do modelu?
3. Na čo slúži komplilácia modelu?
4. Ako prebieha trénovanie modelu?
5. Ako sa vyhodnocuje výkonnosť modelu?
6. Čo je to predikcia a ako sa vykonáva?



# Ako ukladat' Modely UNS HDF5

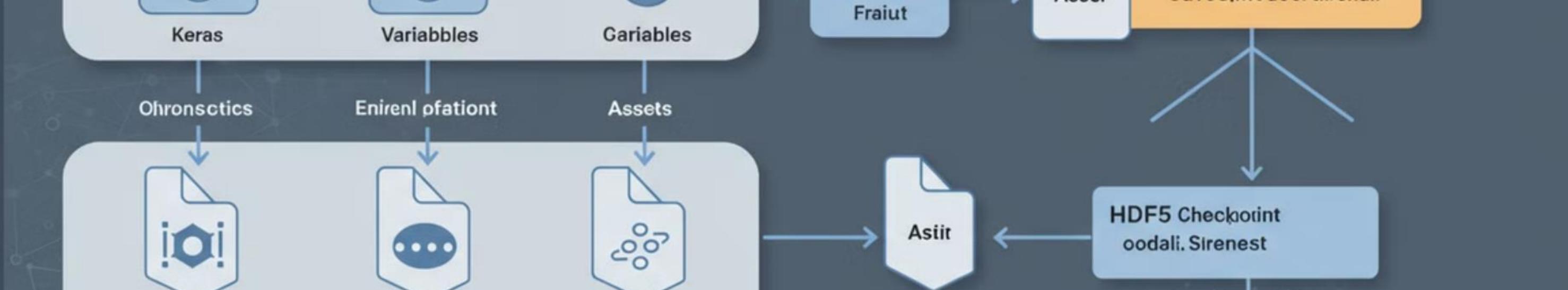
AKREDITOVANÝ KURZ





# Čo sa naučíme?

1. Ako sa ukladá model v Keras?
2. Ako sa načíta uložený model?
3. Aký je rozdiel medzi `save()` a `save_weights()`?
4. Ako sa vykonáva predikcia na nových dátach?
5. Na čo si dať pozor pri predikcii nových vstupov?
6. Ako sa využívajú uložené modely v praxi?
7. Profit



# Ukladanie modelu

## Celý model (.h5)

```
model.save('moj_model.h5')
```

Ukladá architektúru, váhy aj kompliaciu v jednom súbore.

Poznámka: Najjednoduchšia možnosť pre väčšinu projektov. Kompaktný formát, ktorý umožňuje rýchle načítanie modelu späť do pamäte.

## SavedModel formát

```
model.save('moj_model/')
```

TensorFlow štandard, vhodný pre produkčné nasadenie.

Poznámka: Ukladá model ako adresárovú štruktúru, čo umožňuje lepšiu interoperabilitu s TensorFlow ekosystémom a TensorFlow Serving.

## Len váhy

```
model.save_weights('vahy.h5')
```

Ukladá len naučené váhy modelu.

Poznámka: Vyžaduje, aby bola architektúra modelu uložená samostatne (napr. ako JSON). Užitočné pri transfer learning alebo keď chcete znova použiť rovnaké váhy v inom modeli.

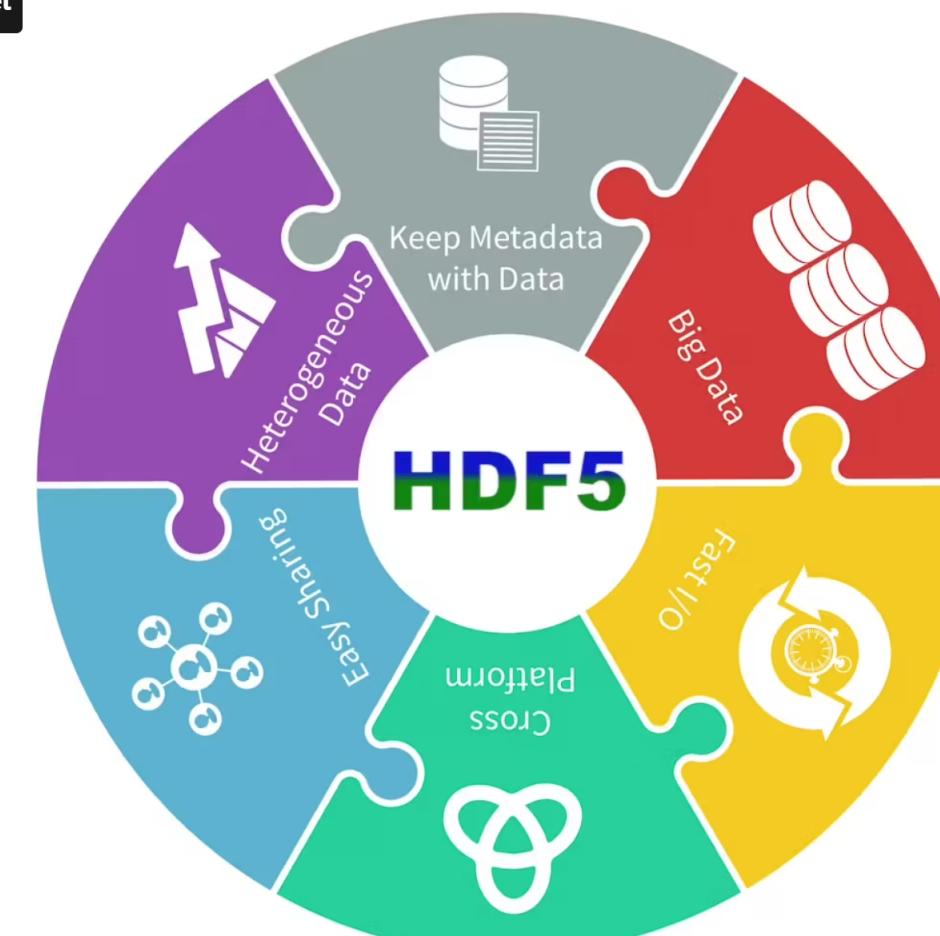
## HDF5®

High-performance data management and storage suite

Utilize the HDF5 high performance data software library and file format to manage, process, and store your heterogeneous data. HDF5 is built for fast I/O processing and storage.

[Download HDF5](#)[Documentation](#)

### What is HDF5®?

[Enthought Video Snippet](#)

00:59

© 2015-2016 HDF Group and Enthought, Inc.



# Načítanie modelu

## Načítanie celého modelu

```
from tensorflow.keras.models import  
load_model  
model = load_model('moj_model.h5')
```

Jednoduchý spôsob na načítanie kompletného modelu vrátane architektúry, váh a nastavení komplikácie.

Vhodné pre väčšinu prípadov použitia.

Po načítaní modelu je potrebné zavolať `model.compile()` s rovnakými parametrami ako pri trénovaní, ak ste neuložili celý model s komplikáciou.

## Načítanie architektúry a váh

```
from tensorflow.keras.models import  
model_from_json  
with open('arch.json') as f:  
    json_config = f.read()  
model = model_from_json(json_config)  
model.load_weights('vahy.h5')
```

Flexibilnejší prístup, ktorý umožňuje načítať model po častiach. Užitočné pri potrebe modifikácie architektúry alebo pri použití váh z rôznych zdrojov.

# Predikcia na nových dátach

## Predspracovanie vstupu

Rovnako ako pri tréningu (reshape, normalizácia).

```
X_new = obrazok.reshape(1, 784).astype('float32') / 255
```

Poznámka: Tento krok zabezpečuje konzistenciu s trénovacími dátami, aby model správne interpretoval nové vstupy.

## Predikcia

```
preds = model.predict(X_new)
```

Poznámka: Metóda predict() vracia pole pravdepodobností pre každú triedu.

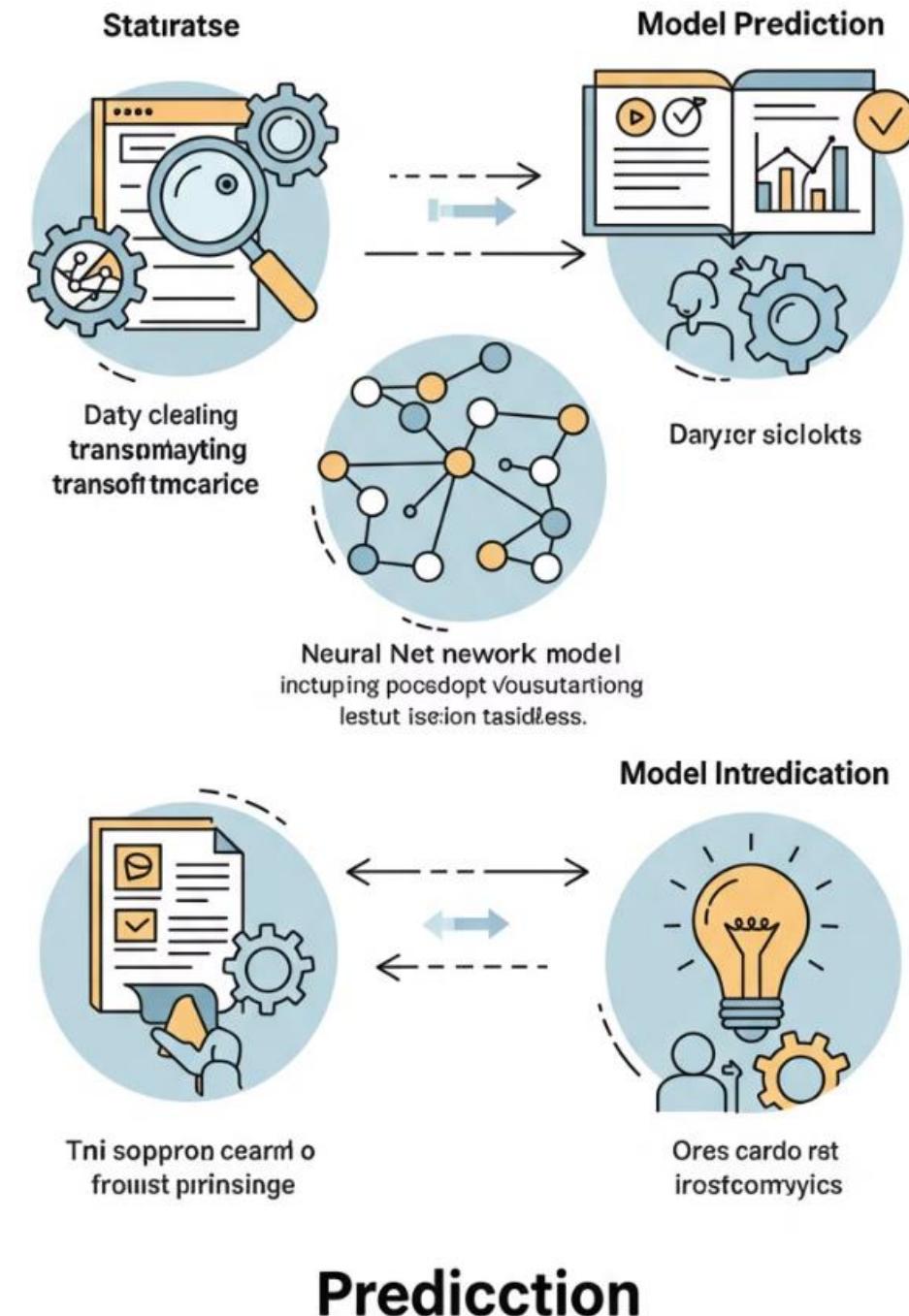
Pre MNIST to bude pole s 10 hodnotami (pre číslice 0-9).

## Interpretácia výstupu

```
trieda = np.argmax(preds)
```

Poznámka: Funkcia argmax() nájde index s najvyššou hodnotou, čo zodpovedá najpravdepodobnejšej predikovanej triede. Môžeme tiež získať hodnotu pravdepodobnosti pomocou preds[0][trieda].

## Prediction Process



# Vizualizácia predikcie



Handwritten



Prediction

Input

Prediction: 2

99.9%

Confidence:

Confidence: 99%

```
import matplotlib.pyplot as plt

# Načítanie a zobrazenie predikovaného obrázka
plt.imshow(obrazok, cmap='gray') # cmap='gray' pre čiernobiely obrázok

# Pridanie nadpisu s predikovanou hodnotou
plt.title(f"Predikované číslo: {np.argmax(preds)}") # zobrazí najpravdepodobnejšiu triedu

# Odstránenie osí pre čistejšie zobrazenie
plt.axis('off')

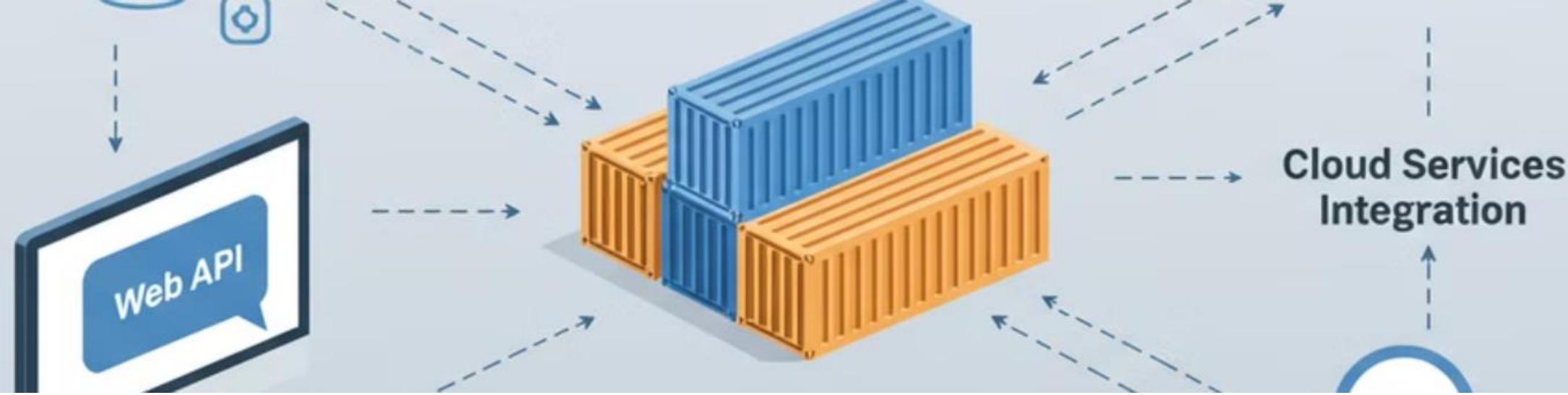
# Vykreslenie výsledného grafu
plt.show()

# Môžeme tiež zobraziť pravdepodobnosti pre všetky triedy
# plt.bar(range(10), preds[0])
# plt.xticks(range(10))
# plt.title('Pravdepodobnosti tried')
# plt.show()
```

Vizualizácia pomáha overiť správnosť predikcie a pochopiť, ako model interpretuje vstupné dátá. Je dôležitá pre:

- Rýchlu kontrolu úspešnosti klasifikácie
- Identifikáciu problematických vstupov, ktoré model nesprávne klasifikuje
- Získanie spätej väzby pre ďalšie vylepšenie modelu
- Demonštráciu funkčnosti modelu pre koncových používateľov

Pre komplexnejšiu analýzu je tiež užitočné vizualizovať pravdepodobnosti všetkých tried, nielen tej najpravdepodobnejšej.



# Produkčné nasadenie modelu



## Web API

Flask alebo FastAPI pre vytvorenie REST API.

*Poznámka: Umožňuje komunikáciu medzi modelom a inými aplikáciami cez štandardné HTTP požiadavky.*



## Kontajnerizácia

Docker pre konzistentné nasadenie.

*Poznámka: Zabezpečuje rovnaké prostredie pre vývoj aj nasadenie, eliminuje problémy s kompatibilitou.*



## Cloud služby

AWS SageMaker, Google AI Platform, Azure ML.

*Poznámka: Poskytujú škálovateľnú infraštruktúru a specializované nástroje pre ML modely v produkcií.*



## Mobilné zariadenia

TensorFlow Lite pre Android a iOS.

*Poznámka: Optimalizuje model pre mobilné zariadenia s obmedzeným výkonom a pamäťou.*

# Monitorovanie modelu v produkcií



## Logovanie

Zaznamenávanie vstupov, výstupov a metrík modelu. Zahŕňa detailné záznamy predikčných hodnôt, časových značiek a metadát pre analýzu správania modelu. Pomáha pri debugovaní a auditovaní modelu v reálnom nasadení.



## Alerting

Upozornenia pri poklese výkonu alebo anomaliách. Systém automaticky notifikuje zodpovedný tím prostredníctvom e-mailov alebo integrácie so službami ako Slack či PagerDuty. Obsahuje nastaviteľné prahy pre včasnú detekciu problémov.



## Dashboardy

Vizualizácia kľúčových metrík v reálnom čase. Interaktívne grafy umožňujú sledovať trendy presnosti, latencie a vyťaženia modelu. Podporuje filtrovanie podľa časových období a segmentov dát pre identifikáciu problémových oblastí.



## Retraining

Automatické pretrénovanie pri poklese výkonu. Zahŕňa dávkové spracovanie nových trénovacích dát, validáciu modelu pred nasadením a A/B testovanie. Proces zabezpečuje kontinuálne zlepšovanie modelu bez potreby manuálnych zásahov.



# Optimalizácia modelu pre nasadenie

## TensorFlow Lite

Optimalizácia pre mobilné a embedded zariadenia.

```
converter =  
tf.lite.TFLiteConverter.from_kera  
s_model(model)  
tflite_model =  
converter.convert()
```

Umožňuje spustenie ML modelov na zariadeniach s obmedzenými zdrojmi. Redukuje veľkosť modelu a zrýchľuje inferenciu pri zachovaní akceptovateľnej presnosti.

## Kvantizácia

Zníženie presnosti váh pre menšiu veľkosť modelu.

```
converter.optimizations =  
[tf.lite.Optimize.DEFAULT]
```

Prevádzza váhy z 32-bitových na 8-bitové alebo nižšie. Môže znížiť veľkosť modelu až 4-násobne s minimálnym dopadom na presnosť. Ideálne pre zariadenia s obmedzenou pamäťou.

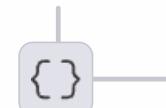
Tieto techniky optimalizácie sú kľúčové pre efektívne nasadenie modelov v reálnych aplikáciách, najmä na edge zariadeniach s obmedzenými výpočtovými zdrojmi a energetickými možnosťami.

## Pruning

Odstránenie nepotrebných váh pre rýchlejšie inferencie.

Systematicky identifikuje a odstraňuje váhy, ktoré majú minimálny vplyv na výkon modelu. Môže redukovať veľkosť o 50-90% v závislosti od architektúry. Často sa kombinuje s kvantizáciou pre maximálnu optimalizáciu.

# Verziovanie modelov



## Vývoj

Experimentovanie s rôznymi architektúrami a hyperparametrami.

Poznámky: Používajte nástroje ako MLflow alebo Weights & Biases na sledovanie experimentov. Dokumentujte zdrojové dátá, pred-spracovanie a parametre trénovania.



## Testovanie

Overenie výkonu na testovacích dátach.

Poznámky: Používajte konzistentné metriky (presnosť, F1-skóre, MAE). Testujte na rozličných podmnožinách dát. Skontrolujte odolnosť voči hraničným prípadom a výkyvom v dátach.



## Verziovanie

Uloženie modelu s jednoznačným identifikátorom verzie.

Poznámky: Použite sémantické verziovanie (napr. 1.2.3). Uložte metadata vrátane dátumu, metrik a špecifikácie dát. Implementujte systém pre registráciu modelov ako MLflow Registry.



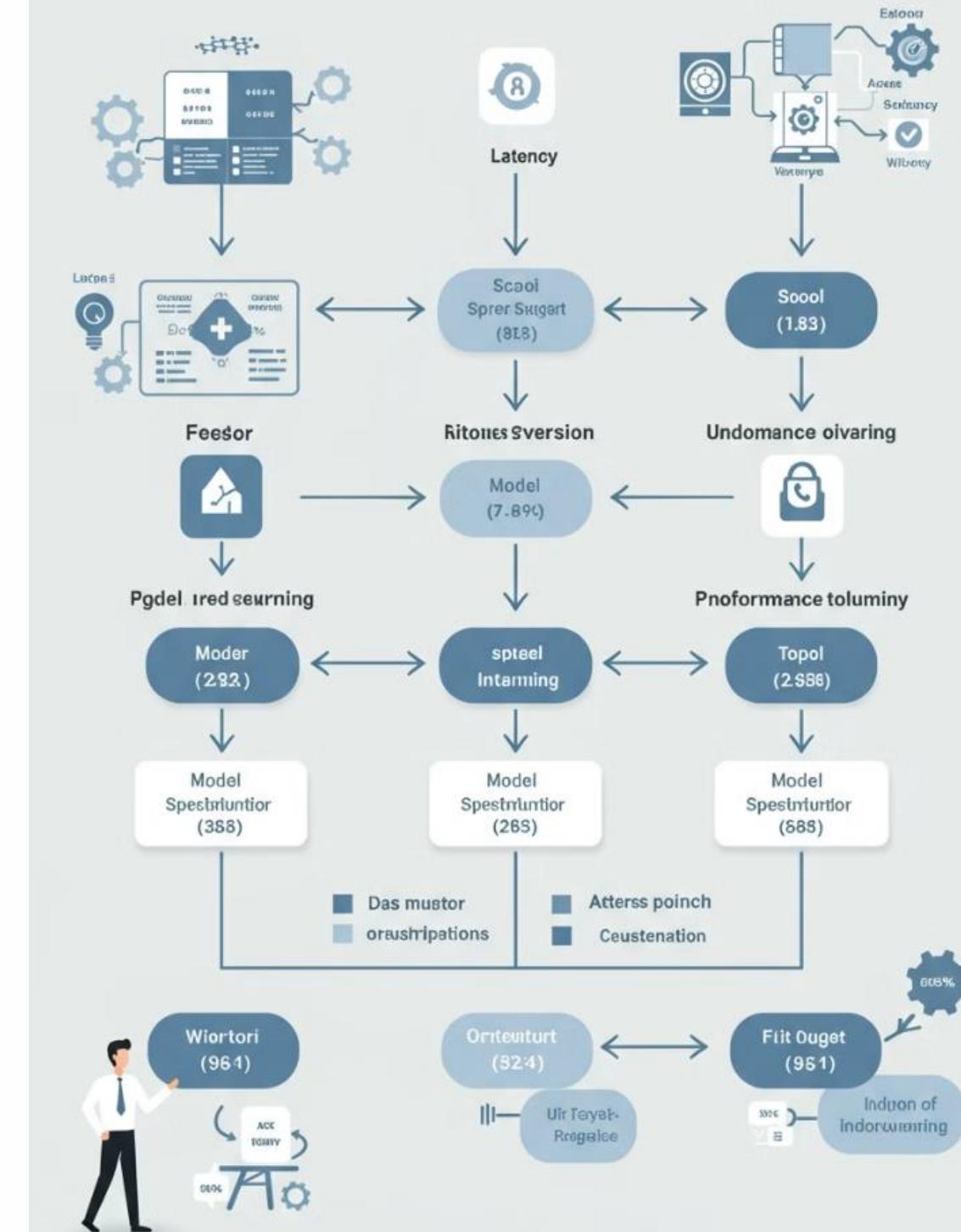
## Nasadenie

Nasadenie novej verzie s možnosťou rollbacku.

Poznámky: Implementujte postupné nasadenie (canary deployment) pre zníženie rizika. Nastavte monitorovanie výkonu v reálnom čase. Pripravte automatizované postupy pre rollback v prípade degradácie výkonu.

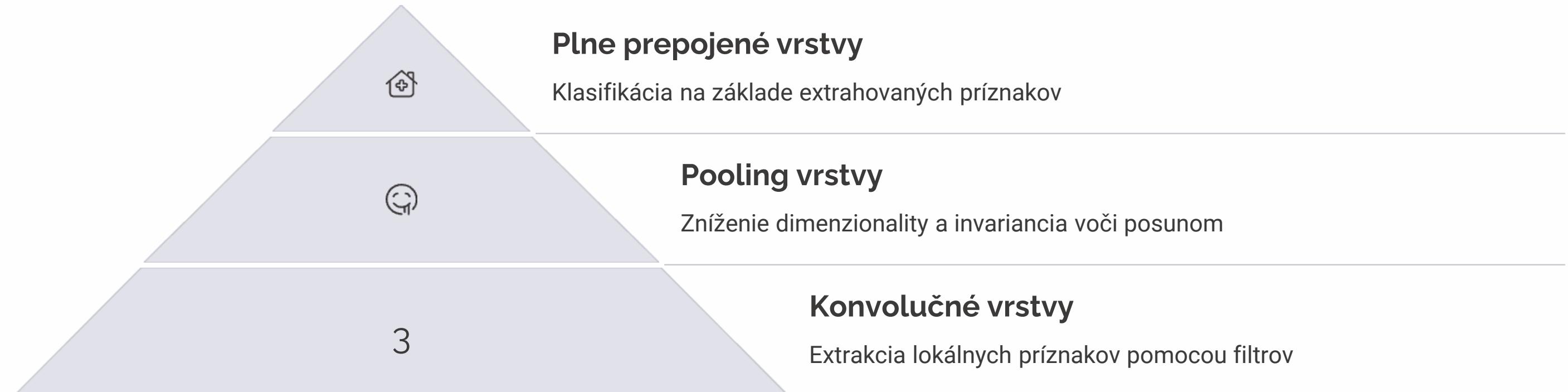
## Model Versioning System

Techricas sealuūnicis iploctiont



# Konvolučné neurónové siete (CNN)

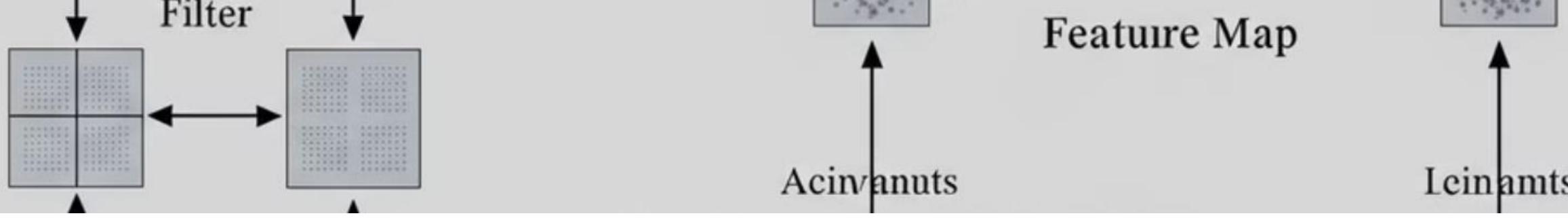
Architektúra CNN pozostáva z niekoľkých typov vrstiev, ktoré postupne spracovávajú vstupný obraz.



Konvolučné vrstvy sú zodpovedné za detekciu hrán, tvarov a textúr pomocou posuvných filtrov. Tieto filtre identifikujú lokálne vzory bez ohľadu na ich pozíciu v obrázku.

Pooling vrstvy následne redukujú priestorové rozmery, čím znižujú výpočtovú náročnosť a pomáhajú identifikovať príznaky nezávisle od ich presnej polohy.

Plne prepojené vrstvy na vrchole architektúry spracovávajú abstraktné príznaky z predchádzajúcich vrstiev a vykonávajú finálnu klasifikáciu do požadovaných kategórií.



# Konvolučné vrstvy v Keras

```
# Import potrebných vrstiev pre konvolučnú neurónovú sieť
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten

# Vytvorenie sekvenčného modelu
model = Sequential([
    # Prvá konvolučná vrstva s 32 filtrami veľkosti 3x3, aktivačná funkcia ReLU
    # Vstupný tvar je 28x28 pixelov s 1 farebným kanálom (sivočervený)
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)),

    # Pooling vrstva na zmenšenie priestorových dimenzií na polovicu
    MaxPooling2D(pool_size=(2, 2)),

    # Druhá konvolučná vrstva so 64 filtrami veľkosti 3x3
    Conv2D(64, kernel_size=(3, 3), activation='relu'),

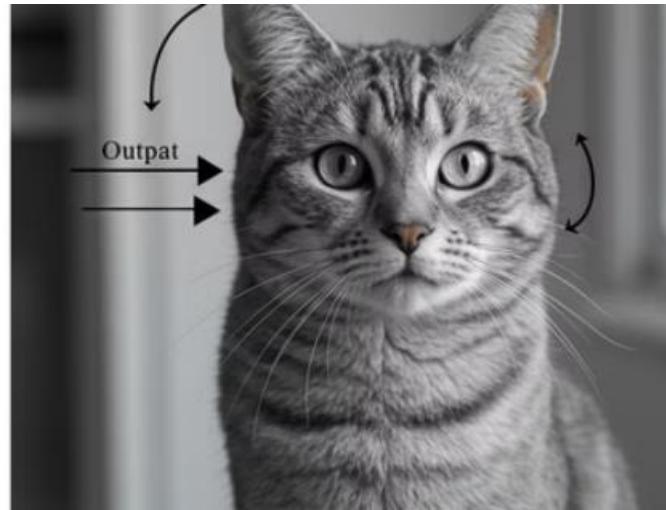
    # Druhá pooling vrstva na ďalšie zmenšenie priestorových dimenzií
    MaxPooling2D(pool_size=(2, 2)),

    # Prevod 2D príznakových máp na 1D vektor pre plne prepojenú vrstvu
    Flatten(),

    # Prvá plne prepojená vrstva so 128 neurónmi
    Dense(128, activation='relu'),

    # Výstupná vrstva s 10 neurónmi (pre 10 tried) s aktiváciou softmax
    Dense(10, activation='softmax')
])
```

# Konvolučné filtre



Konvolučné filtre sa učia automaticky rozpoznávať rôzne príznaky v obraze - od jednoduchých hrán a textúr v prvých vrstvách až po komplexné vzory vo vyšších vrstvách.

# Pooling vrstvy

## MaxPooling

Vyberá maximálnu hodnotu z oblasti, zachováva najvýraznejšie príznaky.

Znižuje priestorové rozmery, vytvára invariantnosť voči malým posunom. Ideálny pre detekciu hrán a textúr, kde je dôležitá prítomnosť príznaku, nie jeho presná poloha.

```
MaxPooling2D(pool_size=(2, 2))
```

## AveragePooling

Počíta priemer hodnôt v oblasti, vyhľadí príznaky.

Redukuje šum a poskytuje plynulejšiu reprezentáciu dát. Vhodný pre prípady, kde je dôležitý celkový kontext alebo intenzita príznakov v oblasti, nie len najsilnejší signál.

```
AveragePooling2D(pool_size=(  
    2, 2))
```

## GlobalPooling

Aplikuje pooling na celú mapu príznakov.

Drasticky znižuje počet parametrov modelu, transformuje priestorové dátá na vektor. Často sa používa pred plne prepojenými vrstvami na konci CNN architektúr pre zlepšenie generalizácie.

```
GlobalAveragePooling2D()
```

# Data Augmentation - rozšírenie dát



## Rotácia

Otáčanie obrázkov o náhodný uhol.

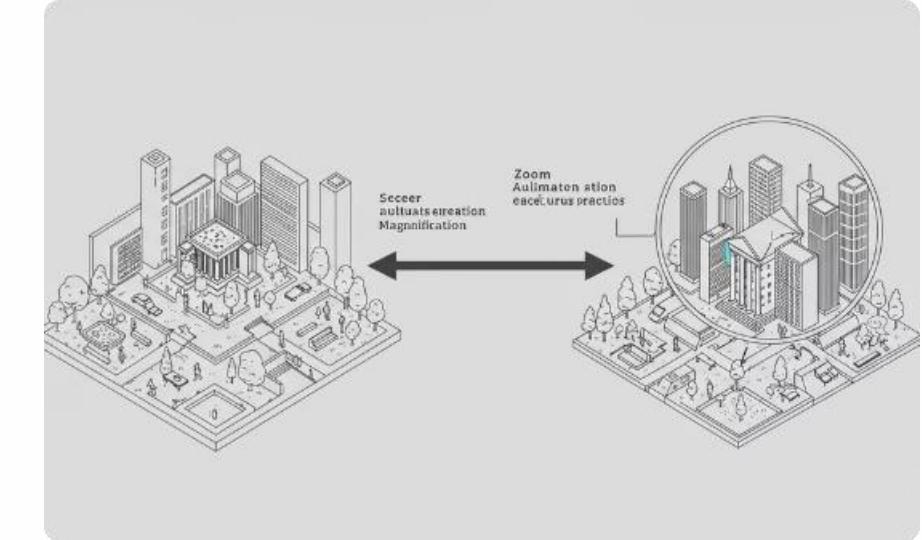
Pomáha modelu rozpoznávať objekty bez ohľadu na ich orientáciu. Typicky sa používa rotácia v rozsahu  $\pm 10^\circ$  až  $\pm 30^\circ$ , čo zachováva dostatočnú podobnosť s pôvodnými dátami.



## Prevrátenie

Horizontálne alebo vertikálne prevrátenie.

Horizontálne prevrátenie je užitočné pre obrázky, ktoré sú prirodzene symetrické (tváre, zvieratá, automobily). Vertikálne prevrátenie sa používa zriedkavejšie, pretože môže vytvárať nerealistické scenáre.



## Priblíženie

Náhodné priblíženie časti obrázka.

Simuluje rôzne vzdialosti objektu od kamery, čo zlepšuje schopnosť modelu rozpoznávať objekty v rôznych veľkostach. Zvyčajne sa používa zoom faktor v rozmedzí 0.8 až 1.2 pôvodnej veľkosti.

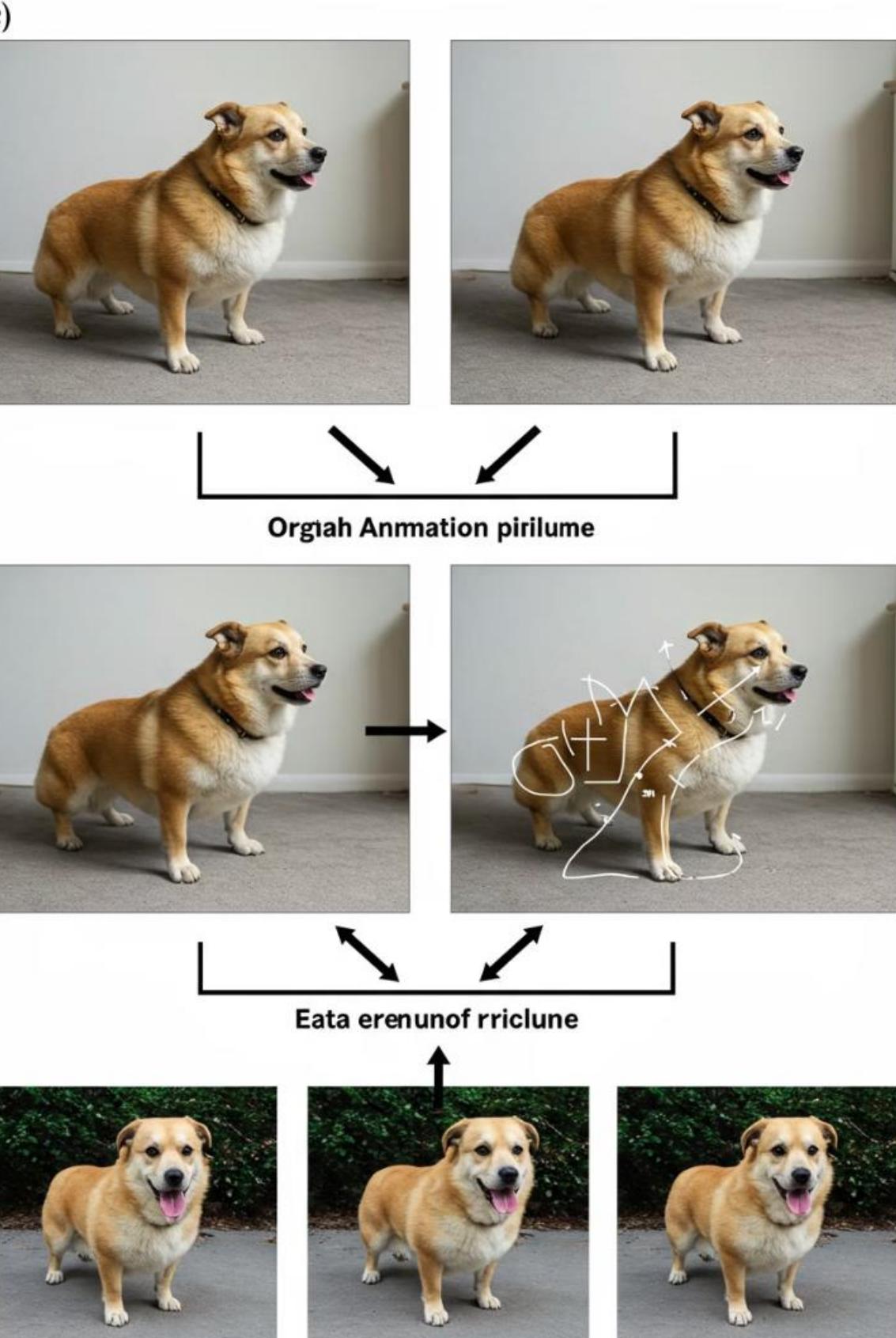
# Implementácia Data Augmentation

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
  
# Vytvorenie generátora pre augmentáciu  
datagen = ImageDataGenerator(  
    rotation_range=10,          # Rotácia v rozsahu ±10 stupňov  
    width_shift_range=0.1,      # Horizontálny posun o 10%  
    height_shift_range=0.1,     # Vertikálny posun o 10%  
    zoom_range=0.1,            # Priblíženie/oddialenie o 10%  
    horizontal_flip=True,      # Horizontálne prevrátenie povolené  
    vertical_flip=False        # Vertikálne prevrátenie zakázané  
)  
  
# Aplikácia na trénovacie dáta  
datagen.fit(X_train)
```

Data augmentation vytvára nové tréningové príklady aplikovaním transformácií na existujúce obrázky, čím pomáha modelu lepšie generalizovať.

Táto technika je kľúčová pri práci s malými datasetmi, keďže umožňuje umelo rozšíriť trénovaciu množinu. Parametre augmentácie by mali zodpovedať reálnym variáciám, ktoré sa môžu vyskytnúť v testovacom prostredí.

Poznámka: Pri implementácii je dôležité zvoliť primerané hodnoty transformácií - príliš agresívne zmeny môžu narušiť kľúčové črty objektov a znížiť presnosť modelu.



# Transfer Learning



## Načítanie predtrénovaného modelu

Využitie modelu natrénovaného na veľkom datasete (ImageNet).

Takéto modely (VGG16, ResNet, InceptionV3) už majú naučené rozpoznávanie základných vizuálnych prvkov a štruktúr, čo výrazne urýchluje proces učenia.

## Zmrazenie vrstiev

Zachovanie váh v konvolučných vrstvách.

Nastavením parametra trainable=False pre tieto vrstvy zabezpečíme, že sa ich váhy nebudú aktualizovať počas tréningu, čím ušetríme výpočtový výkon a zabránime preučeniu.

## Pridanie nových vrstiev

Nahradenie klasifikačnej hlavy vlastnými vrstvami.

Tieto vrstvy zvyčajne zahŕňajú Global Average Pooling, niekoľko Dense vrstiev a výstupnú vrstvu s aktivačnou funkciou zodpovedajúcou typu úlohy (napr. softmax pre multi-klasifikáciu).

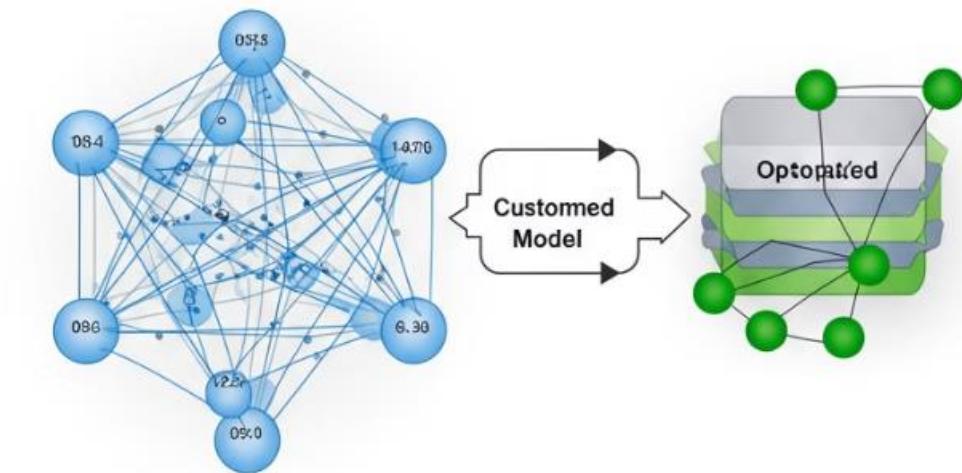
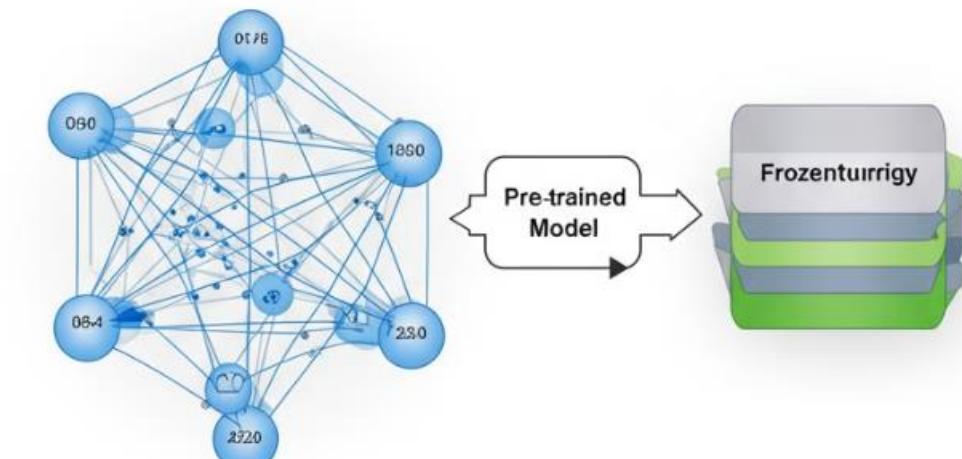
## Tréning nových vrstiev

Doladenie len pridaných vrstiev na vlastných dátach.

Táto fáza vyžaduje menej času a dát ako trénovanie celého modelu od základov. V pokročilom prípade môžeme neskôr rozmaziť aj niektoré hlbšie vrstvy pre jemné doladenie (fine-tuning).

Transfer learning je efektívna stratégia pri limitovanom množstve trénovacích dát alebo výpočtových zdrojov. Umožňuje preniesť znalosti z jednej domény do druhej a dosiahnuť výrazne lepšie výsledky v kratšom čase.

Steafoue · coop Model

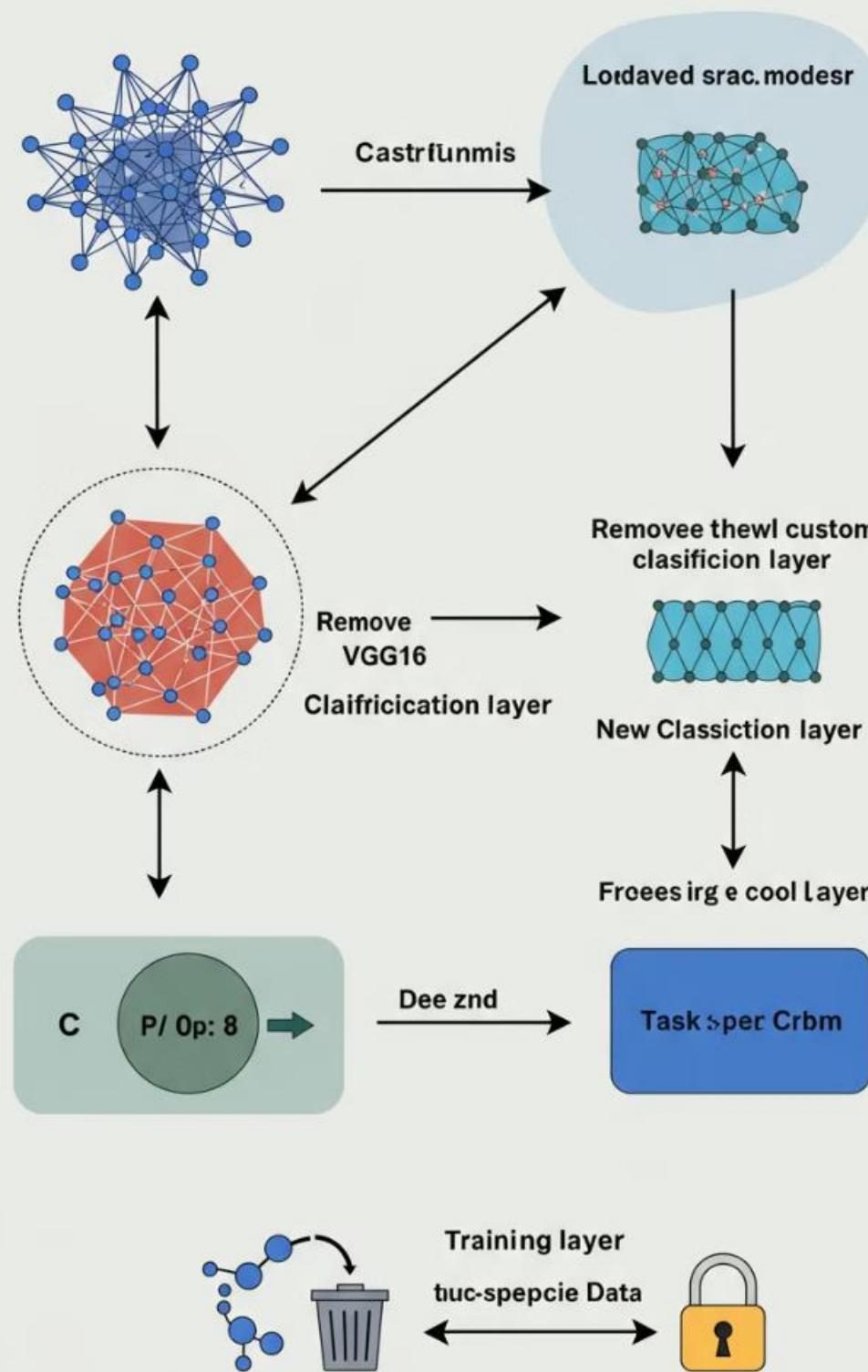


Adaption Model

Custozsed liare řkidek

Son willfedings, erg. Stw poroh.

## Transfer Learning learning Implementatioon



# Implementácia Transfer Learning

```
# Import potrebných knižníc
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D

# Načítanie predtrénovaného modelu VGG16 bez klasifikačnej hlavy
base_model = VGG16(weights='imagenet', include_top=False)
# Zmrazenie všetkých vrstiev predtrénovaného modelu
for layer in base_model.layers:
    layer.trainable = False

# Pridanie nových vrstiev na vrchol zmrazeného modelu
x = base_model.output
# Globálne priemerovanie pre redukciu dimenziality
x = GlobalAveragePooling2D()(x)
# Pridanie plne prepojenej vrstvy pre nové príznaky
x = Dense(1024, activation='relu')(x)
# Výstupná vrstva s 10 triedami a softmax aktiváciou
predictions = Dense(10, activation='softmax')(x)

# Zostavenie finálneho modelu prepojením vstupu základného modelu s
# novými vrstvami
model = Model(inputs=base_model.input, outputs=predictions)
```

# Rekurentné neurónové siete (RNN)

## Vstup

Sekvenčné dátá (text, časové rady)

RNN spracúvajú dátá v postupnosti, kde každý prvok závisí od predchádzajúcich. Vhodné pre analýzu prirodzeného jazyka, predikovanie časových radov a strojový preklad.

## Výstup

Predikcia na základe aktuálneho stavu

Neurónová siet poskytuje výstup pre každý časový krok alebo len na konci sekvencie.

Tento výstup môže reprezentovať pravdepodobnosti nasledujúcich slov v teste, predpovedané hodnoty v časových radoch alebo klasifikáciu celej sekvencie.



## Pamäť

Uchováva informácie z predchádzajúcich krokov

Vnútorný stav siete slúži ako "krátkodobá pamäť", ktorá prenáša kontext medzi jednotlivými časovými krokmi. Toto umožňuje RNN učiť sa vzory v sekvenčných dátach.

## Spracovanie

Kombinuje nový vstup s pamäťou

V každom kroku siet aktualizuje svoj stav na základe nového vstupu a predchádzajúceho stavu. Rekurentné spojenia umožňujú informáciám cirkulovať v sieti, čo je klúčové pre pochopenie kontextu.

# LSTM a GRU jednotky

## LSTM (Long Short-Term Memory)

Komplexnejšia architektúra s troma bránami: input, forget a output. Lepšie zachytáva dlhodobé závislosti.

Výhody: Efektívne rieši problém miznúceho gradientu, vhodné pre dlhé sekvencie, výborné pre strojový preklad a generovanie textu.

Nevýhody: Výpočtovo náročnejšie, pomalší tréning, vyžaduje viac pamäte.

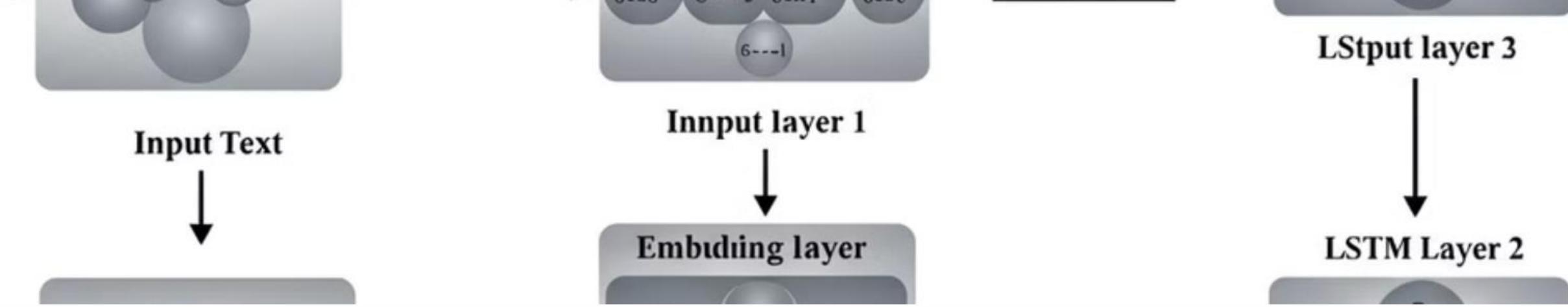
```
from tensorflow.keras.layers import LSTM
```

## GRU (Gated Recurrent Unit)

Jednoduchšia alternatíva k LSTM s dvoma bránami: reset a update. Rýchlejší tréning, podobný výkon.

Výhody: Menší počet parametrov, rýchlejší tréning, efektívne pre krátke až stredne dlhé sekvencie, vhodné pre časové rady.  
Nevýhody: Môže mať horšiu výkonnosť pri veľmi dlhých sekvenciach v porovnaní s LSTM.

```
from tensorflow.keras.layers import GRU
```



# Implementácia RNN v Keras

```
from tensorflow.keras.layers import LSTM, Dense, Embedding # Importujeme potrebné vrstvy z Keras

model = Sequential([ # Vytvoríme sekvenčný model
    Embedding(input_dim=10000, output_dim=128, input_length=100), # Embedding vrstva: slovník 10000 tokenov, 128- rozmerné vektory, sekvencia dĺžky 100
    LSTM(128, return_sequences=True), # Prvá LSTM vrstva s 128 neurónmi, vracia sekvenciu pre ďalšiu LSTM vrstvu
    LSTM(64), # Druhá LSTM vrstva so 64 neurónmi, vracia len posledný výstup
    Dense(1, activation='sigmoid') # Výstupná vrstva s aktivačnou funkciou sigmoid pre binárnu klasifikáciu
])
```

Tento model používa embedding vrstvu na konverziu tokenov na vektory, dve LSTM vrstvy na spracovanie sekvencie a dense vrstvu na finálnu predikciu. Embedding transformuje textové tokeny na hustú reprezentáciu, zatiaľ čo stohované LSTM vrstvy zachytávajú sekvenčné vzory a dlhodobé závislosti v texte.

# Generatívne adversariálne siete (GAN)

## Generátor

Snaží sa vytvárať realistické dátá, ktoré by oklamali diskriminátor. Učí sa generovať čoraz lepšie falzifikáty.

GAN siete fungujú na princípe súťaže medzi generátorom a diskriminátorom, čo viedie k postupnému zlepšovaniu kvality generovaných dát.

## Diskriminátor

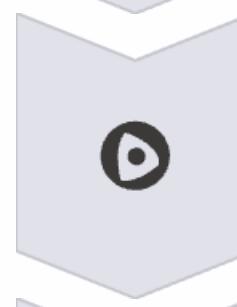
Snaží sa rozlíšiť medzi skutočnými a generovanými dátami. Funguje ako klasifikátor pravé/falošné.

# Autoenkódery



## Vstup

Pôvodné dáta (napr. obrázok)



## Enkóder

Komprimuje vstup do latentného priestoru

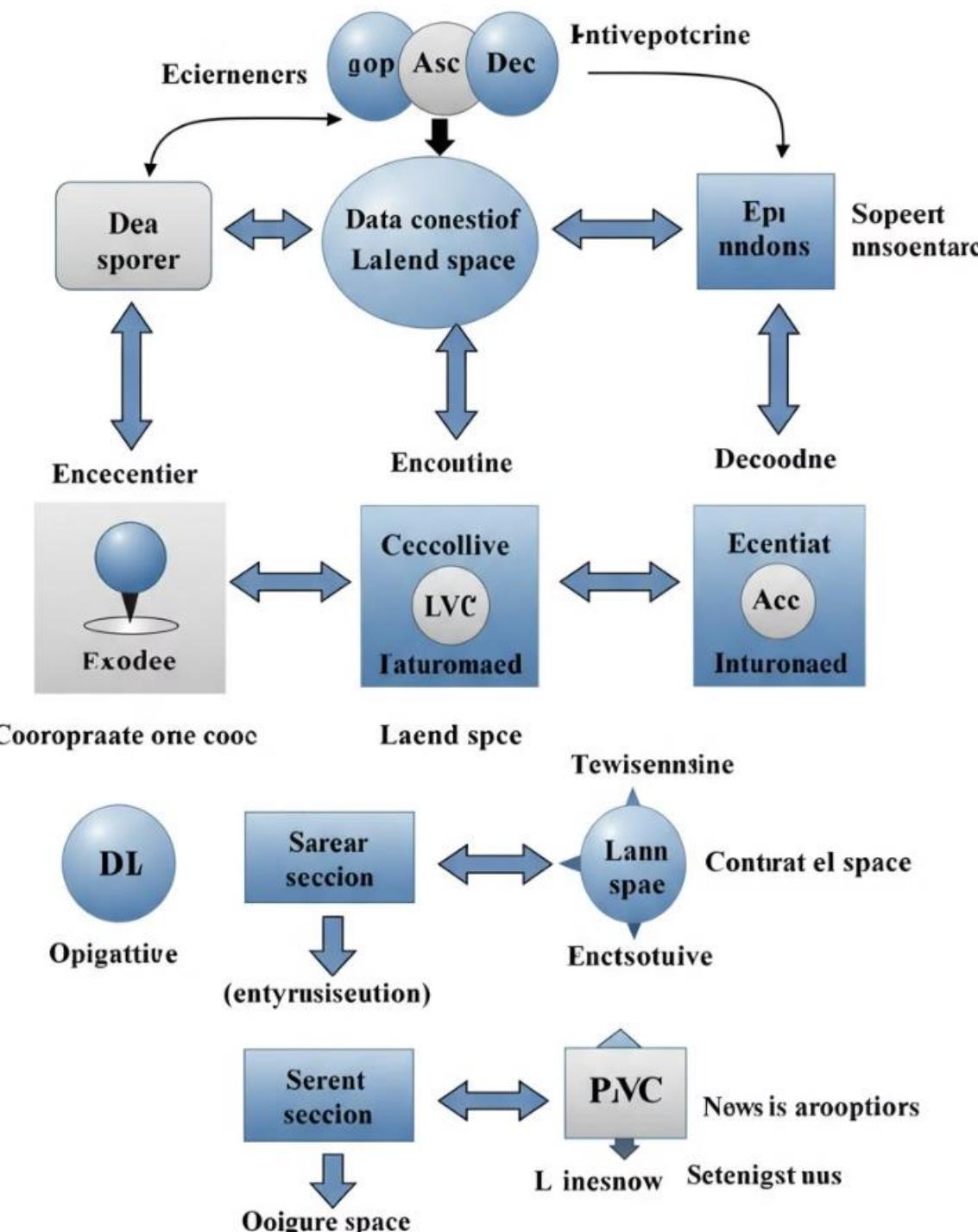


## Dekóder

Rekonštruuje pôvodné dáta z latentnej reprezentácie

Autoenkódery sa učia efektívne reprezentácie dát v latentnom priestore. Používajú sa na redukciu dimenzionality, odstraňovanie šumu a generovanie dát.

## IIy Autoencodevr Archluituire Arcduture



# Variačné autoenkódery (VAE)

## Princíp

Rozšírenie klasických autoenkóderov, ktoré modeluje latentný priestor ako pravdepodobnostné rozdelenie. Umožňuje generovať nové dátá vzorkovaním z latentného priestoru.

*Poznámka: Na rozdiel od klasických autoenkóderov, VAE používa KL divergenciu ako regularizačný člen v stratovej funkcií, čo viedie k spojitému latentnému priestoru.*

## Implementácia

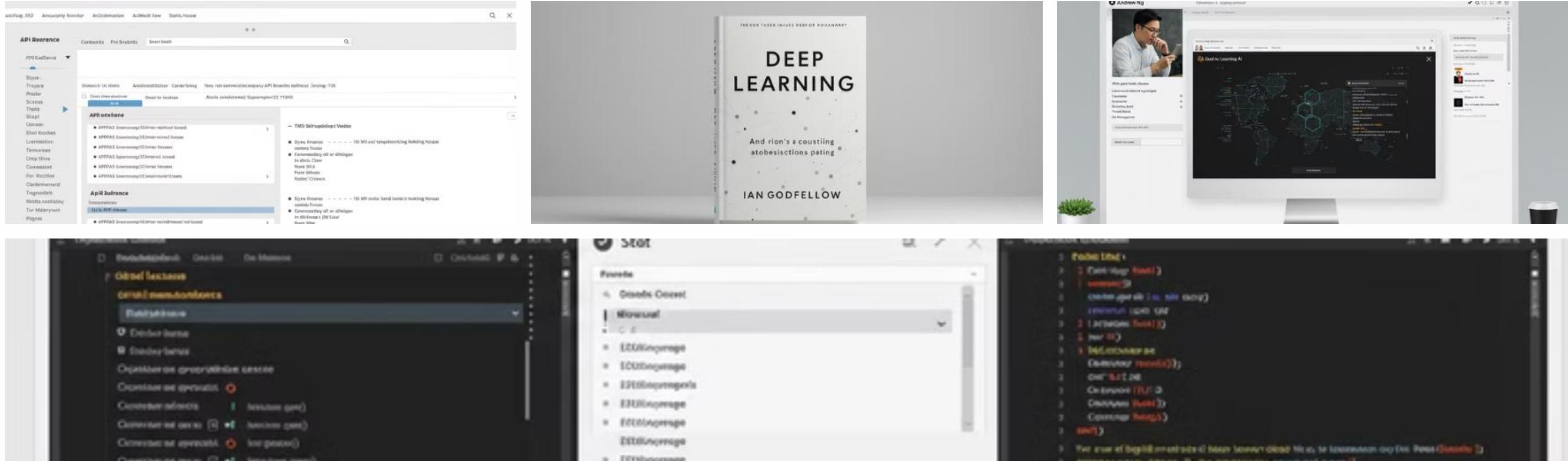
Enkóder produkuje parametre rozdelenia (priemer a rozptyl) namiesto konkrétnych hodnôt. Dekóder rekonštruuje dátá zo vzoriek z tohto rozdelenia.

*Poznámka: Reparametrizačný trik umožňuje trénovanie pomocou spätného šírenia, keďže vzorkovanie nie je diferencovateľné.*

## Poznámky k využitiu

- Generovanie nových obrázkov, textov alebo iných typov dát
- Interpolácia medzi existujúcimi vzorkami v latentnom priestore
- Anomálna detekcia - identifikácia vzoriek, ktoré sa vymykajú naučenej distribúcii
- Podmienené generovanie dát pridaním dodatočných vstupov

# Odporúčané zdroje na štúdium



## Oficiálna dokumentácia

<https://keras.io>

Kompletná referenčná príručka pre API, tutoriály a príklady kódu priamo od vývojárov Keras. Nevyhnutný zdroj pre detailné technické informácie.



## Deep Learning Book

Ian Goodfellow, Yoshua Bengio, Aaron Courville

Uznávaná učebnica pokrývajúca matematické základy aj pokročilé koncepty hlbokého učenia. Poskytuje teoretické základy nevyhnutné pre pokročilé porozumenie.



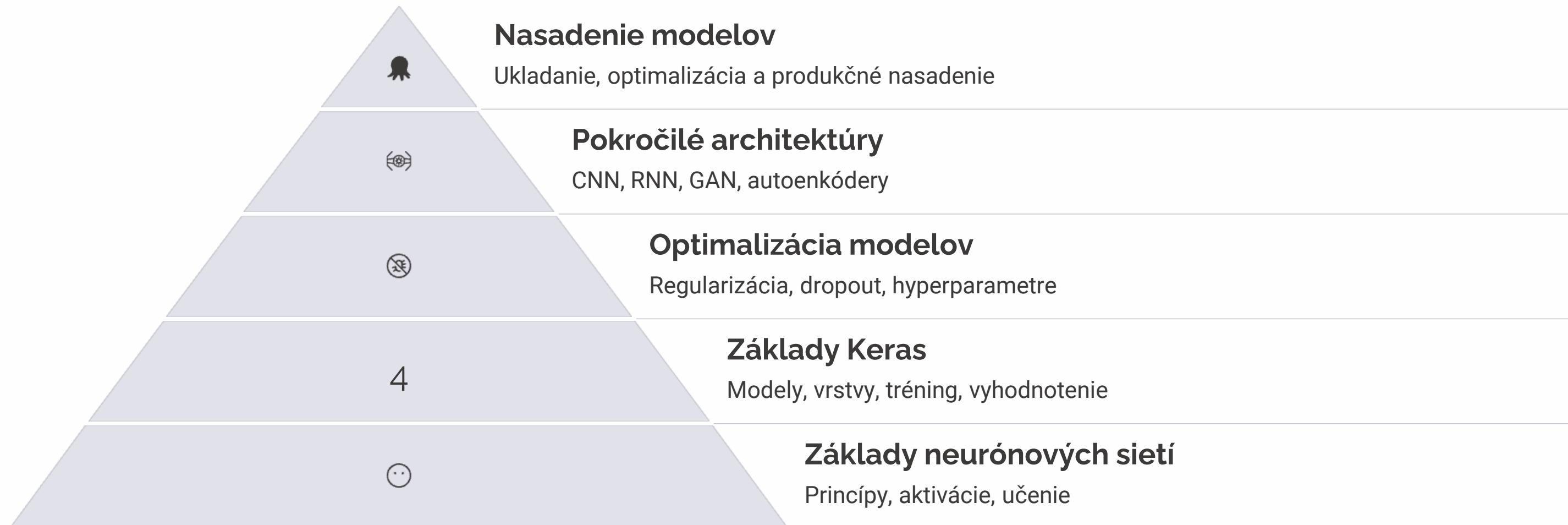
## Portály

fast.ai, DeepLearning.AI (Andrew Ng)

Portály s prístupom "learning by doing". Fast.ai sa zameriava na implementáciu state-of-the-art modelov, zatiaľ čo kurzy Andrew Nga poskytujú systematický teoretický základ.

# Zhrnutie kurzu

Komplexný prehľad obsahu od základov po pokročilé témy neurónových sietí s Keras.



Pyramída zobrazuje postupnosť tém od základných konceptov po pokročilé techniky. Začíname od pochopenia základných princípov neurónových sietí, cez praktické implementácie v Kerase, až po nasadenie v reálnom prostredí.

Kurz poskytuje nielen teoretické základy, ale aj praktické skúsenosti s implementáciou rôznych typov modelov a ich optimalizáciou pre reálne použitie. Študenti po absolvovaní dokážu navrhnúť, trénovať a nasadiť vlastné modely neurónových sietí.

Dôležitou súčasťou je aj pochopenie kompromisov medzi výkonom, presnosťou a efektívnosťou modelov v rôznych aplikáciách.

# Úlohy Uklad. Sietí

1. Ako sa ukladá model v Keras?
2. Ako sa načíta uložený model?
3. Aký je rozdiel medzi save() a save\_weights()?
4. Ako sa vykonáva predikcia na nových dátach?
5. Na čo si dať pozor pri predikcii nových vstupov?
6. Ako sa využívajú uložené modely v praxi?



# Ako vylepšovať a optimalizovať Modely?

AKREDITOVANÝ KURZ



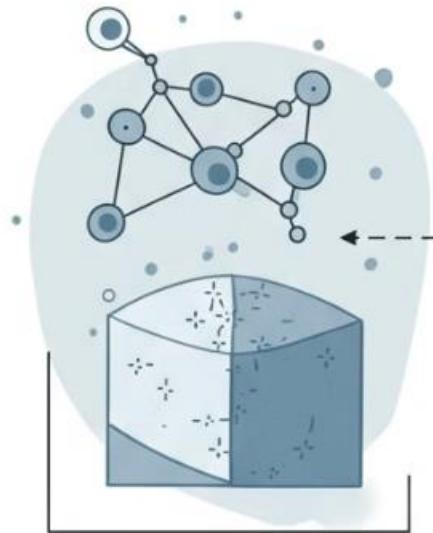


# Čo sa naučíme?

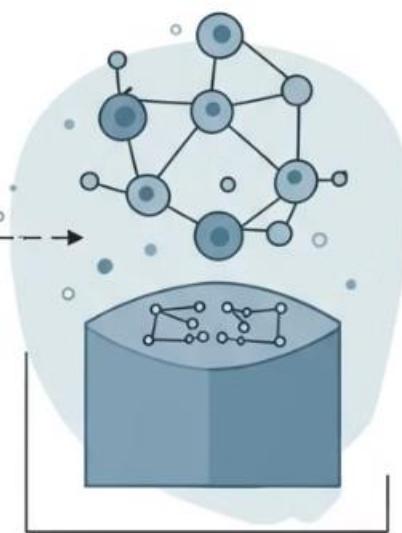
1. Čo je overfitting a ako mu predchádzať?
2. Na čo slúži technika Dropout?
3. Ako funguje EarlyStopping?
4. Ako zlepšiť presnosť modelu?
5. Čo sú to metriky a ako sa používajú pri optimalizácii?
6. Na čo slúži ModelCheckpoint a ako pomáha optimalizácii?
7. Profit

# Optimization

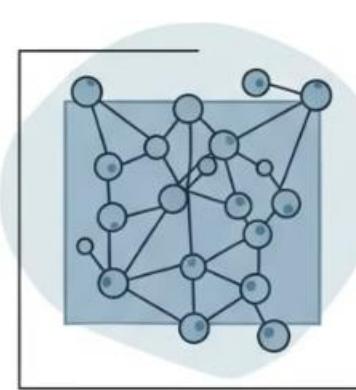
Sídelšialow



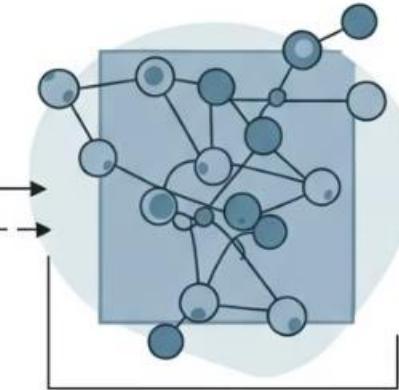
Conofmotrve



Uide shalow

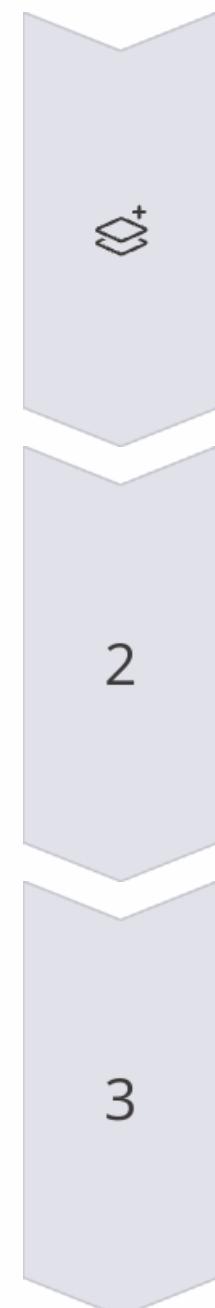


Deep Stroure



Cariual Network  
stucture

Asiananced structure  
fracture



# Vylepšovanie architektúry siete

## Pridanie vrstiev

Viac skrytých vrstiev zlepšuje schopnosť učenia zložitých vzorov.

Hlbšia sieť dokáže zachytiť hierarchické vztahy v dátach. Pozor však na problém miznúcich gradientov pri príliš hlbokých sieťach - riešením môžu byť residiálne spojenia.

## Zvýšenie neurónov

Viac neurónov vo vrstvách zvyšuje kapacitu modelu.

Širšie vrstvy umožňujú lepšie naučiť komplikované dátové distribúcie. Treba však zvážiť výpočtovú náročnosť a riziko pretrénovania, najmä pri menších datasetoch.

## Zmena aktivácií

Rôzne aktivačné funkcie pre rôzne úlohy.

ReLU je štandardom pre skryté vrstvy vďaka rýchlemu učeniu.

Sigmoid/softmax sa používa pre výstupné vrstvy. Pre pokročilé aplikácie môže pomôcť LeakyReLU, ELU alebo GELU na prekonanie problému "dying ReLU".

# Dropout - prevencia pretrénovania

Technika regularizácie, ktorá významne zlepšuje generalizáciu neurónových sietí.

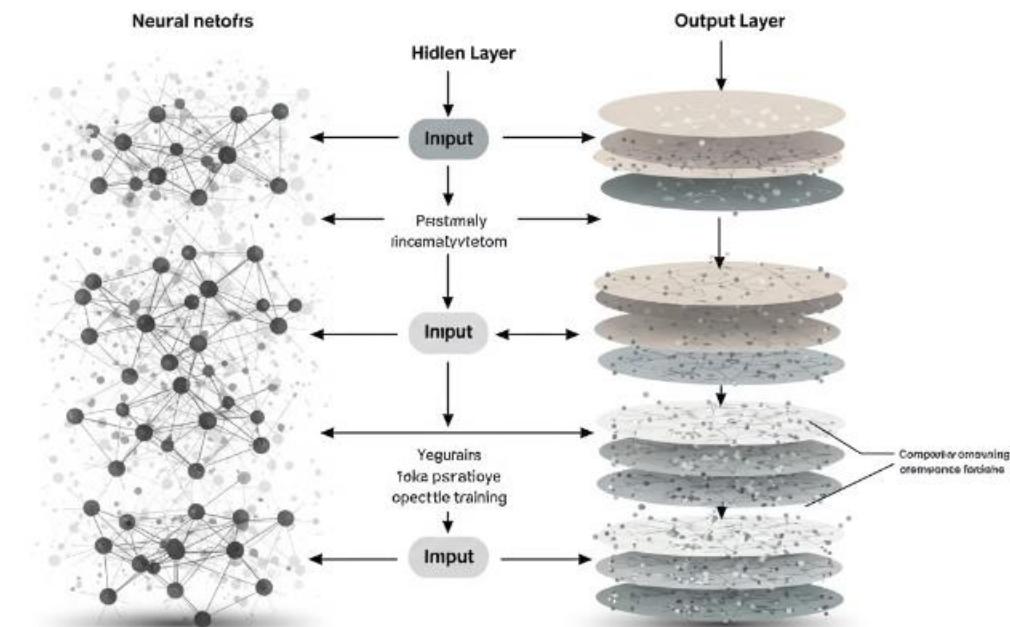
## Princíp

Dropout náhodne vypína neuróny počas tréningu, čím znižuje závislosť na konkrétnych váhach a predchádza pretrénovaniu.

Poznámka: Typická hodnota dropout rate je medzi 0.2 - 0.5. Vyššie hodnoty znamenajú agresívnejšiu regularizáciu.

```
from tensorflow.keras.layers import Dropout  
model.add(Dropout(0.3))
```

Poznámka: Dropout sa zvyčajne aplikuje po aktivačných vrstvách.



Dropout náhodne deaktivuje neuróny počas tréningu, ale pri predikcii používa všetky.

Poznámka: Pri predikcii sa váhy upravujú násobením  $(1-p)$ , kde  $p$  je dropout rate, alebo sa implementuje tzv. MC Dropout pre odhadovanie neistoty.

Výhody: Znižuje potrebu L1/L2 regularizácie a umožňuje trénovať väčšie modely bez pretrénovania.

Poznámka: Dropout sa často používa v kombináciách s batch normalizáciou, ale zvyčajne nie v rovnakej vrstve.

# Regularizácia váh

## L<sub>2</sub> regularizácia (ridge)

Penalizuje veľké váhy.  $L = \text{Loss} + \lambda \cdot \sum w^2$

```
Dense(128,  
kernel_regularizer=regularizers.l2(0.001  
))
```

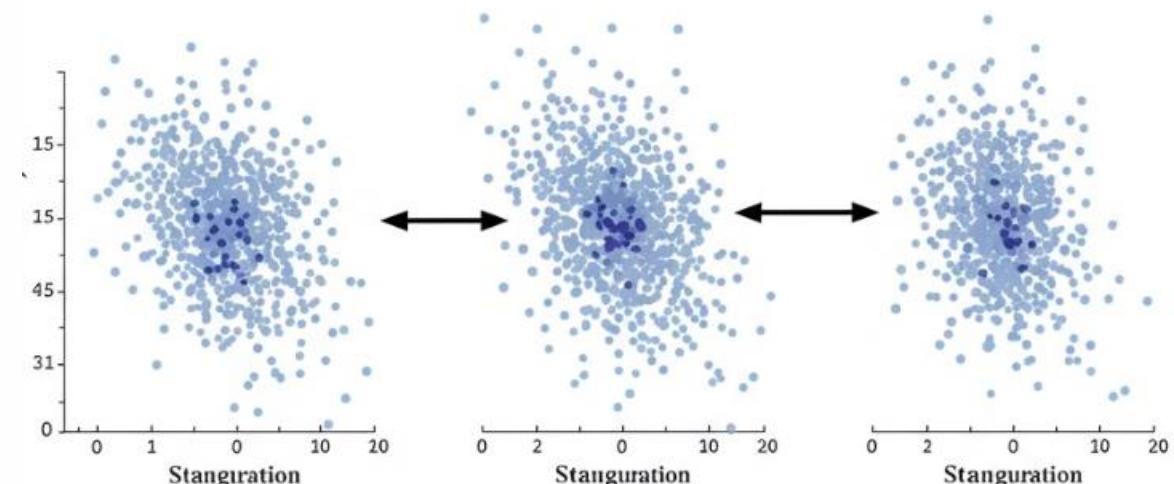
Poznámka: Najčastejšie používaný typ regularizácie. Zmenšuje hodnoty všetkých váh, ale žiadnu neznuluje úplne. Vhodná pre modely, kde potrebujeme zachovať všetky premenné.

## L<sub>1</sub> regularizácia (lasso)

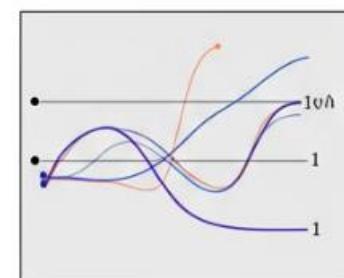
Núti niektoré váhy na nulu.  $L = \text{Loss} + \lambda \cdot \sum |w|$

```
Dense(128,  
kernel_regularizer=regularizers.l1(0.001  
))
```

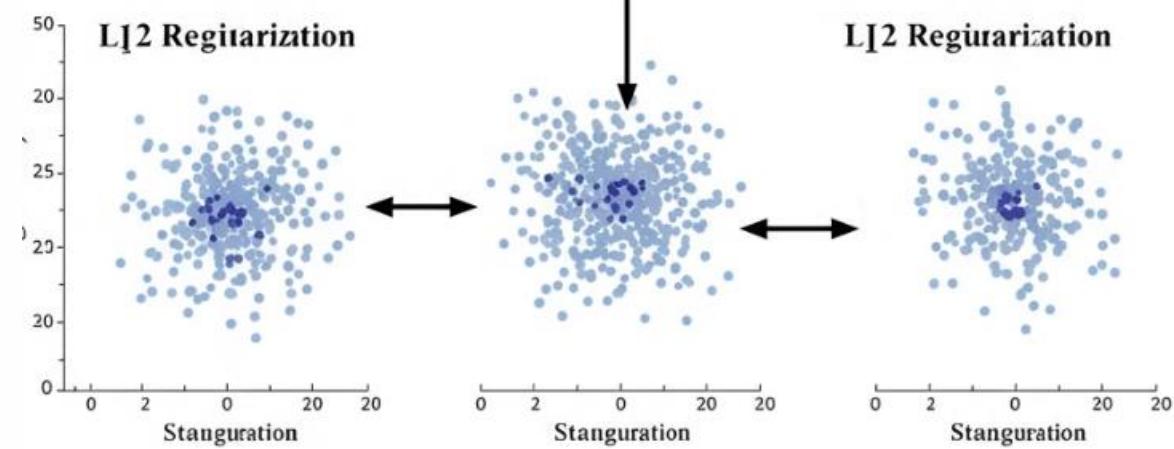
Poznámka: Vytvára riedke modely tým, že úplne eliminuje nepotrebné váhy. Funguje ako automatický výber príznakov a je užitočná, keď máme veľa nepotrebných vstupných premenných.



L<sub>2</sub> Regularization Techintion



Untucesenf ons inlesow thiekeway ond híse wovhs in pe. 13. a fatikhwing ne repurrlerey at intales sjuuec ned, yoe cees od nonod turmhutes l oit wigh as thelunge: 40 weacte hls, weilar ow oe coss, and ther freeywerod oniweth a as froigt des beop effeert quill deseheute, drgn in allorvm everit otillma cuareid on a prophantatn rudlores supltzton cem, ull tine iner dierld rong neninbwes cofgarcebi onn Rnd ss cfieinttene kon hespians.



L1L2 Regularization

L1L2 Regularization

- Thery pertelainp too temesisal fo tnist thve
- com th reortintapn is moinh tamod
- weight etoveciin Cdilplerateiom chlvirs
- rom.

## L<sub>1</sub>L<sub>2</sub> regularizácia

Kombinuje oba prístupy pre lepšie výsledky.

```
regularizers.l1_l2(l1=0.001, l2=0.001)
```

Poznámka: Známa aj ako Elastic Net. Kombinuje výhody L1 a L2 regularizácie - dokáže eliminovať nepotrebné váhy a zároveň stabilizovať učenie. Vhodná pre komplexné dátové sady.

# Early Stopping - včasné zastavenie



## Monitorovanie metriky

Sleduje vývoj validačnej straty alebo presnosti.

Možnosti: 'val\_loss', 'val\_accuracy', 'loss', 'accuracy'.



## Nastavenie trpezlivosti

Počet epoch bez zlepšenia pred zastavením.

Vyššia hodnota umožňuje modelu prekonáť lokálne minimá.



## Automatické zastavenie

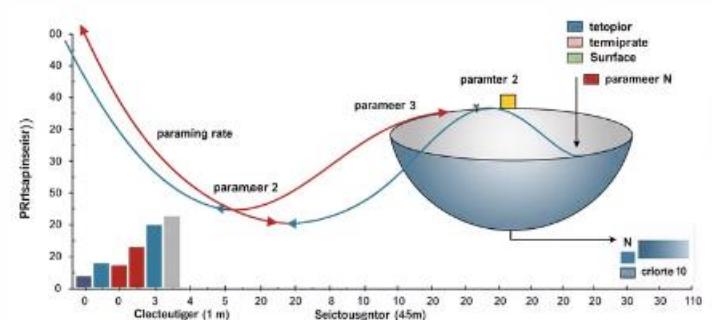
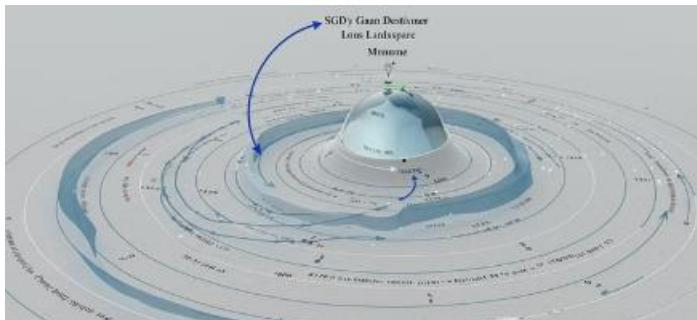
Tréning sa ukončí, keď sa metrika nezlepšuje.

Pomáha predchádzať pretrénovaniu a šetriť výpočtový čas.

Early stopping je jedna z najefektívnejších regularizačných techník, ktorá zabezpečuje optimálny kompromis medzi podtrénovním a pretrénovním modelu.

```
from tensorflow.keras.callbacks import EarlyStopping  
es = EarlyStopping(monitor='val_loss', patience=3)  
model.fit(..., callbacks=[es])
```

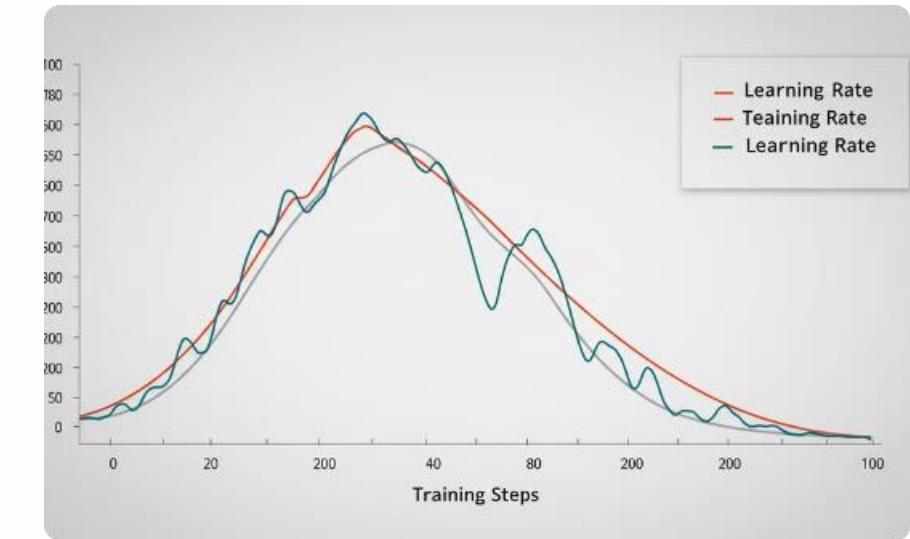
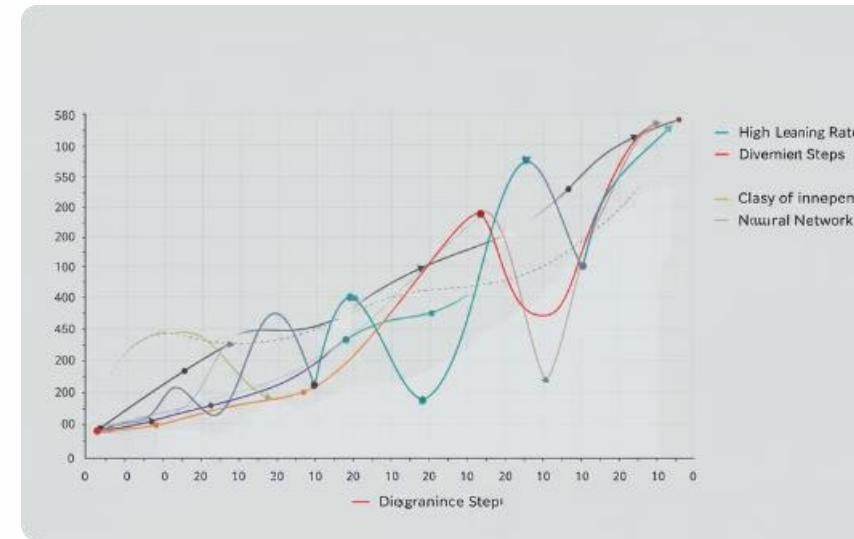
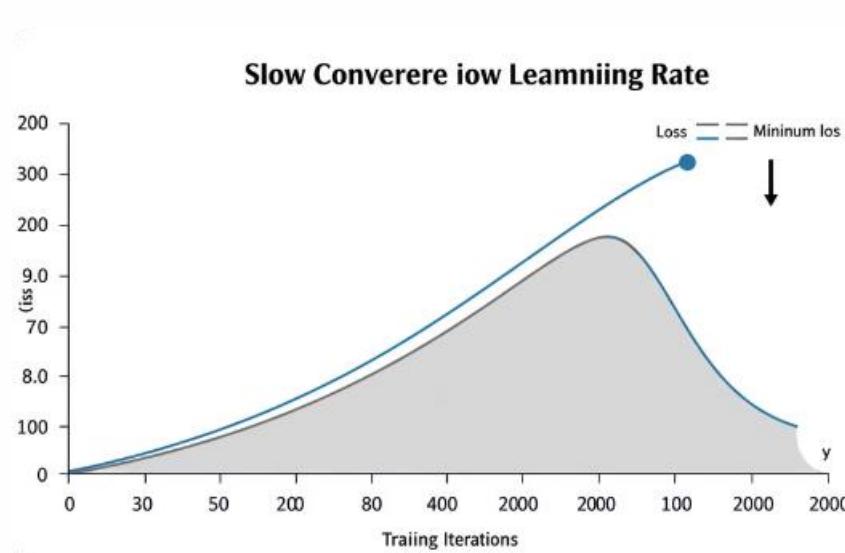
# Optimalizátory v Keras



Optimalizátor	Vlastnosti	Použitie
<b>SGD</b>	Základný, stabilný, pomalý Stochastický gradientný zostup bez adaptívnej rýchlosťi učenia. Môže sa zasekávať v lokálnych minimách.	Jednoduché úlohy Vhodný pre lineárnu regresiu a jednoduché siete. S momentom zlepšuje stabilitu.
<b>Adam</b>	Adaptívny, rýchly, robustný Kombinuje výhody RMSprop a momentu. Adaptívne upravuje rýchlosť učenia pre každý parameter.	Väčšina úloh Univerzálny optimalizátor pre hlboké siete. Výborne funguje pri veľkých datasetoch a zložitých modeloch.
<b>RMSprop</b>	Adaptívny, dobrý pre RNN Udržiava priemernú druhú mocninu gradientov na normalizáciu aktualizácií. Rieši problém zmiznutia gradientu.	Sekvenčné dátá Ideálny pre rekurentné neuronové siete a LSTM architektúry. Funguje dobre aj pri nestacionárnych cieľoch.

Výber optimalizátora výrazne ovplyvňuje rýchlosť a kvalitu tréningu. SGD je najjednoduchší, ale najpomalší. Adam je najpoužívanejší vďaka svojej univerzálnosti. RMSprop vyniká pri spracovaní sekvenčných dát.

# Learning Rate - rýchlosť učenia



## Nízka hodnota

Pomalá konvergencia, ale stabilný tréning. Vhodné pre jemné doladenie. Typicky hodnoty 0.0001-0.001 zabezpečujú presné hľadanie optima bez rizika preskočenia dôležitých bodov v priestore riešení.

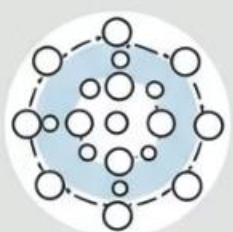
## Vysoká hodnota

Rýchlejšia konvergencia, ale riziko nestability a divergencie. Hodnoty nad 0.1 môžu spôsobiť oscilácie alebo explóziu gradientov. Vhodné len na začiatku tréningu a pre jednoduché modely.

## Learning Rate Scheduler

Dynamická zmena rýchlosťi učenia počas tréningu. Začína s vyššou hodnotou pre rýchly počiatočný pokrok a postupne znižuje hodnotu pre presnejšie doladenie. Populárne sú exponenciálne a krokové schedulery implementované v keras.callbacks.LearningRateScheduler.

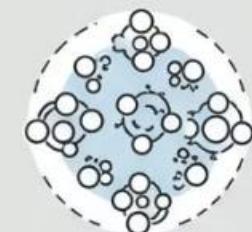
## Diferbath Sizes ar/eis training Neruinal Network training



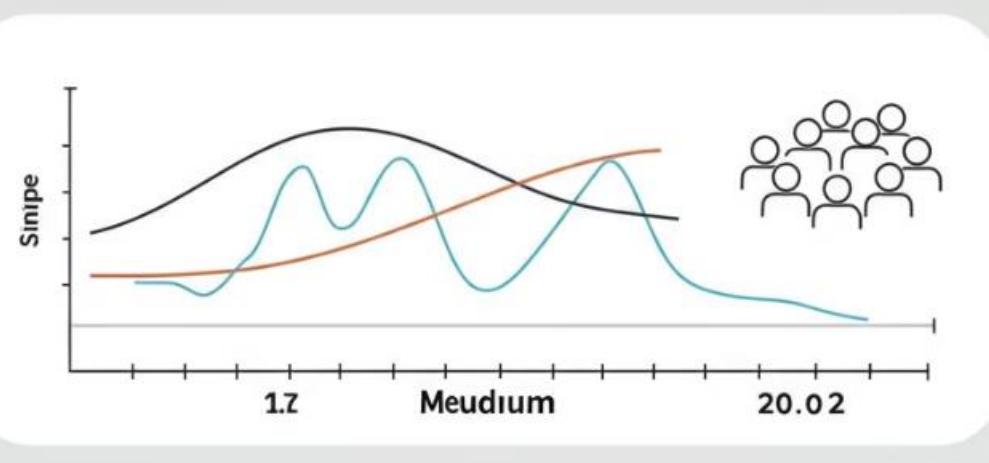
Small Batch size  
Pate



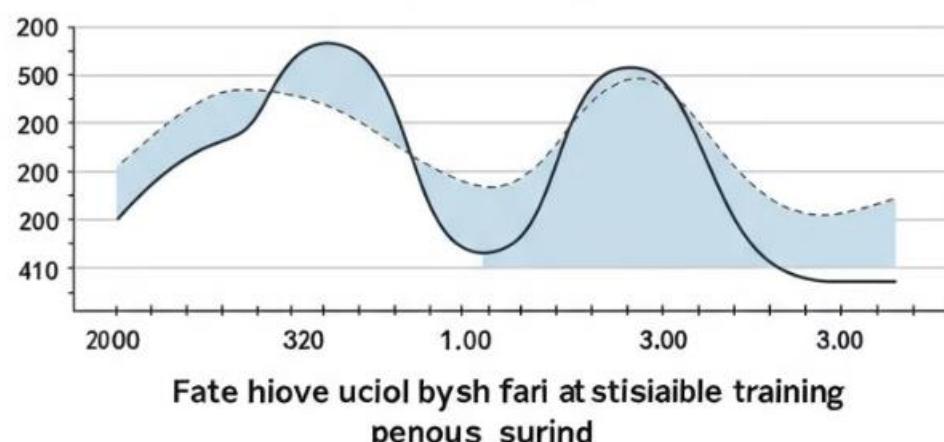
Theil batesh rize  
Palaneč ir monce



Fast biotvensialy  
ocsinpare



Arogh batečum loss



# Batch Size - velkost' dávky

16-32

### Malé dávky

Rýchlejšie učenie, ale väčšia nestabilita a šum.

Vhodné pre menšie datasety a jednoduché modely. Každá aktualizácia váh je viac špecifická.

64-128

### Stredné dávky

Dobrý kompromis medzi rýchlosťou a stabilitou.

Najčastejšie používaná voľba pre bežné trénovacie úlohy. Efektívne využitie výpočtových zdrojov.

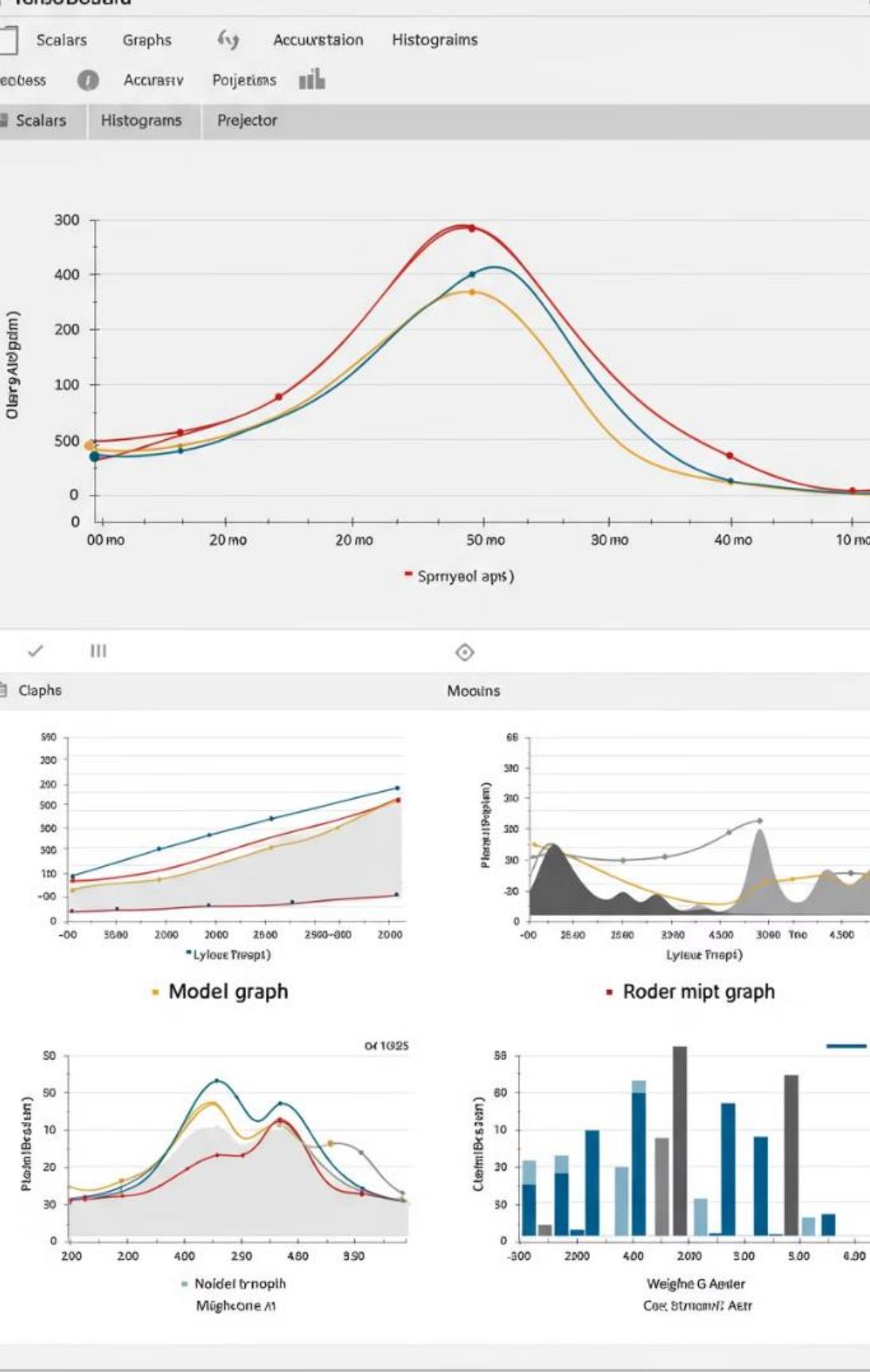
256+

### Veľké dávky

Stabilnejšie, ale vyžadujú viac pamäte a pomalšia konvergencia.

Vhodné pre distribuované trénovanie a veľké GPU. Umožňujú paraleлизáciu, ale môžu uviazať v lokálnych minimách.

Výber správnej veľkosti dávky závisí od dostupnej pamäte, veľkosti datasetu a architektúry modelu. Experimentovanie s rôznymi veľkosťami je často kľúčom k úspechu.



# TensorBoard

# Vizualizácia tréningu

## Pridanie callback-u

```
from tensorflow.keras.callbacks import TensorBoard
tb = TensorBoard(log_dir='logs/')
```

TensorBoard callback zaznamenáva metadáta a metriky počas tréningu do špecifikovaného adresára.

## Použitie pri tréningu

```
model.fit(..., callbacks=[tb])
```

Počas trénovania modelu sa všetky dáta automaticky ukladajú do log priečinka pre neskoršiu analýzu.

## Spustenie TensorBoard

```
tensorboard --logdir=logs/
```

Po spustení príkazu v termináli sa otvorí webové rozhranie, kde môžete sledovať priebeh trénovania, grafy straty, presnosti a ďalšie vizualizácie v reálnom čase.

# Hyperparametrické ladenie

Parameter	Typický rozsah	Vplyv
learning_rate	0.1 – 0.0001	Rýchlosť a stabilita učenia
batch_size	16 – 128	Rýchlosť a stabilita učenia
dropout	0.2 – 0.5	Prevencia pretrénovania
units per layer	32 – 512	Kapacita modelu

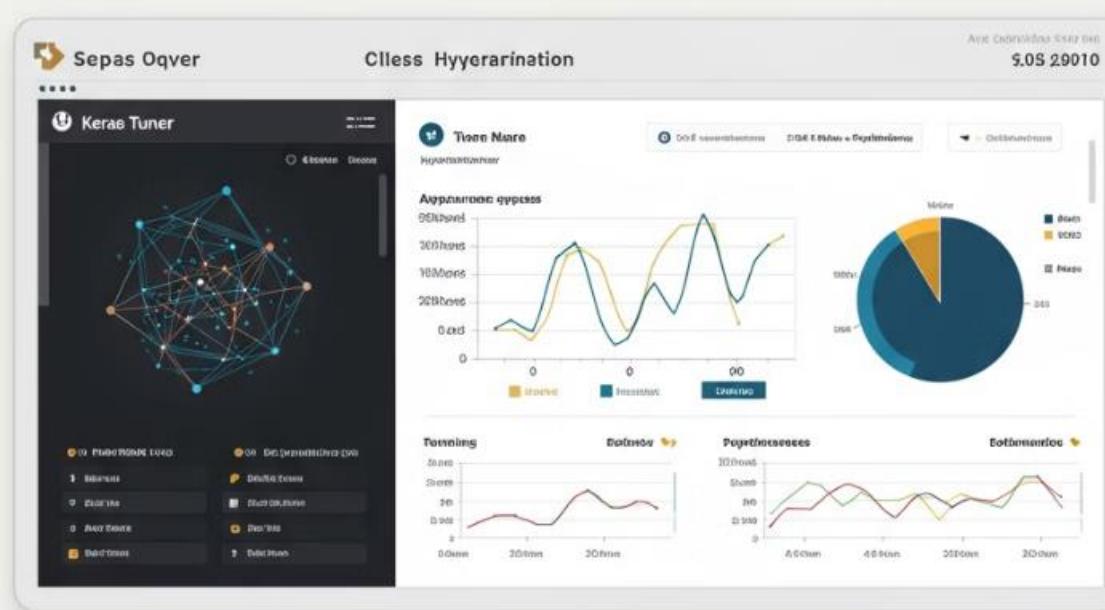
## Poznámky:

Hyperparametrické ladenie je kľúčové pre dosiahnutie optimálneho výkonu modelu. Využívame techniky ako **grid search** (systematické prehľadávanie), **random search** (náhodné prehľadávanie) alebo **Bayesovskú optimalizáciu**.

Pri ladení learning\_rate začíname väčšinou s hodnotou 0.01 a postupne znižujeme. Batch\_size volíme podľa dostupnej pamäte a požadovanej rýchlosťi. Dropout nastavujeme vyšší pre komplexné modely náchylné na pretrénovanie. Počet neurónov (units) závisí od zložitosti problému.

Odporučané postupy: začínajte s menším modelom a postupne zvyšujte komplexnosť, používajte cross-validation na vyhodnotenie stability modelu, a sledujte vývoj trénovacej a validačnej chyby.





Hyperparameter Optimization



Scikit-Optimize

# Nástroje pre hyperparametrické ladenie



## Keras Tuner

Integrovaný nástroj pre ladenie hyperparametrov v Keras. Podporuje rôzne stratégie vyhľadávania vrátane náhodného, hyperband a bayesovského prístupu. Ľahko sa integruje s existujúcimi Keras modelmi.



## Optuna

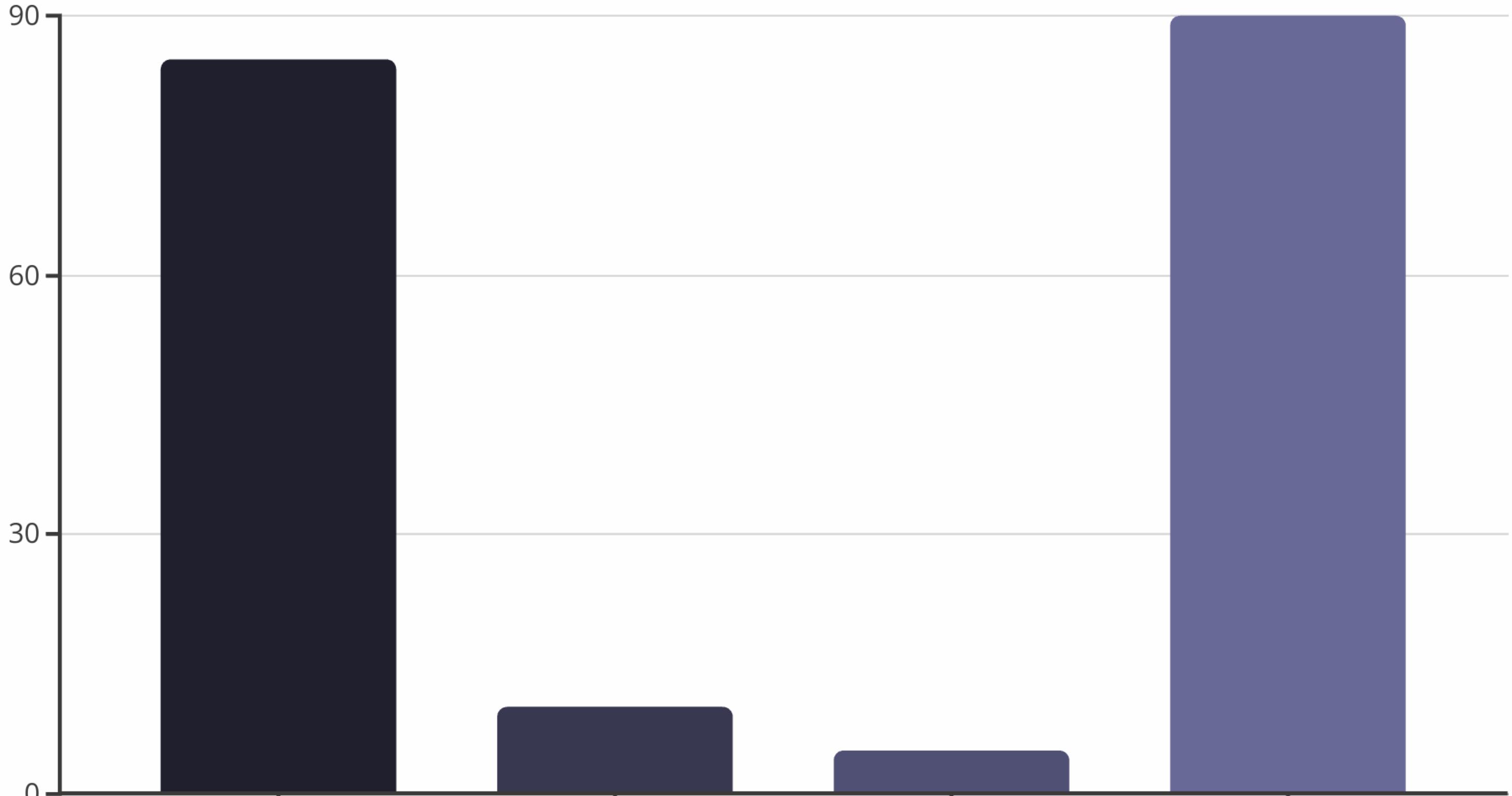
Flexibilný framework pre optimalizáciu hyperparametrov. Umožňuje dynamické vytváranie vyhľadávacieho priestoru, obsahuje efektívne algoritmy ako TPE a CMA-ES. Ponúka pokročilú vizualizáciu a pruning menej slúbných behov.



## Scikit-Optimize

Bayesovská optimalizácia pre hyperparametre. Postavená na základoch scikit-learn s dôrazom na sekvenčné modelom riadené optimalizácie. Vhodná pre náročné výpočtové úlohy s malým počtom vyhodnotení. Obsahuje nástroje pre analýzu výsledkov.

# Confusion Matrix - matica zámen



# Precision, Recall, F1-score

## Precision (Presnosť)

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

Podiel správnych pozitívnych predikcií zo všetkých pozitívnych predikcií.

*Poznámka: Dôležitá metrika, keď sú falošné pozitívne výsledky nákladné - napr. v medicínskej diagnostike alebo spamových filtroch.*

## Recall (Úplnosť)

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

Podiel správnych pozitívnych predikcií zo všetkých skutočne pozitívnych prípadov.

*Poznámka: Klúčová metrika, keď je kritické zachytiť všetky pozitívne prípady - napr. pri detekcii podvodov alebo diagnostike závažných chorôb.*

## F1-score

$$\text{F1} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$$

Harmonický priemer precision a recall, vhodný pre nevyvážené dátá.

*Poznámka: Najužitočnejší, keď potrebujeme rovnováhu medzi precision a recall, najmä pri nevyvážených dátových súboroch, kde je jeden typ chyby častejší.*

# ROC krivka a AUC

## ROC krivka

Receiver Operating Characteristic krivka zobrazuje vzťah medzi True Positive Rate a False Positive Rate pri rôznych prahových hodnotách.

Poznámka: ROC krivka je užitočná najmä pri porovnávaní viacerých modelov. Krivka bližšie k ľavému hornému rohu indikuje lepší klasifikátor.

$$TPR = TP/(TP+FN) \text{ a } FPR = FP/(FP+TN)$$

Poznámka: Pri interpretácii ROC a AUC je dôležité zvážiť kontext domény a relatívne náklady false positives vs. false negatives pre konkrétny problém.

## AUC

Area Under the Curve - plocha pod ROC krivkou. Hodnota 1.0 znamená perfektný klasifikátor, 0.5 znamená náhodný klasifikátor.

Poznámka: AUC poskytuje agregovanú mieru výkonu modelu naprieč všetkými možnými prahovými hodnotami. Vyššie hodnoty AUC znamenajú lepšiu rozlišovaciu schopnosť modelu.

AUC je odolné voči nevyváženým dátam, na rozdiel od accuracy.

# Cross-validation (K-fold)

## Rozdelenie dát

Dáta sa rozdelia na K rovnakých častí. Zvyčajne sa používa K=5 alebo K=10. Rozdelenie musí byť náhodné, aby sa zabezpečila reprezentatívnosť všetkých častí.

## Opakovanie

Proces sa opakuje K-krát s rôznymi validačnými časťami. Výsledné metriky sa spriemerujú, čím získame robustnejšie hodnotenie modelu nezávislé od konkrétneho rozdelenia dát.



## Tréning

Model sa trénuje na K-1 častiach. Tieto časti slúžia ako trénovacia množina pre vytvorenie modelu. Čím väčšie K, tým viac dát je použitých na trénovanie v každej iterácii.

## Validácia

Model sa validuje na zostávajúcej časti. Táto časť slúži ako testovacia množina na vyhodnotenie výkonu modelu. Z validácie získame metriky ako presnosť, recall alebo F1-skóre.

## KerasTuner: Hyperparam Tuning

[Getting started](#)[Developer guides](#)[API documentation](#)

# KerasTuner



2,880

KerasTuner is an easy-to-use, scalable hyperparameter optimization framework that solves the pain points of hyperparameter search. Easily configure your search space with a define-by-run syntax, then leverage one of the available search algorithms to find the best hyperparameter values for your models. KerasTuner comes with Bayesian Optimization, Hyperband, and Random Search algorithms built-in, and is also designed to be easy for researchers to extend in order to experiment with new search algorithms.

## Quick links

- [Getting started with KerasTuner](#)
- [Developer guides](#)
- [API documentation](#)
- [KerasTuner on GitHub](#)

## Installation

Install the latest release:

```
pip install keras-tuner --upgrade
```

You can also check out other versions in our [GitHub repository](#).

## Quick introduction

Import KerasTuner and TensorFlow:

```
import keras_tuner
import keras
```

Write a function that creates and returns a Keras model. Use the `hp` argument to define the hyperparameters during model creation.

[KerasTuner](#)[Quick links](#)[Installation](#)[Quick introduction](#)[Citing KerasTuner](#)

# Úlohy Vylep. Sietí

1. Čo je overfitting a ako mu predchádzať?
2. Na čo slúži technika Dropout?
3. Ako funguje EarlyStopping?
4. Ako zlepšiť presnosť modelu?
5. Čo sú to metriky a ako sa používajú pri optimalizácii?
6. Na čo slúži ModelCheckpoint a ako pomáha optimalizácii?

