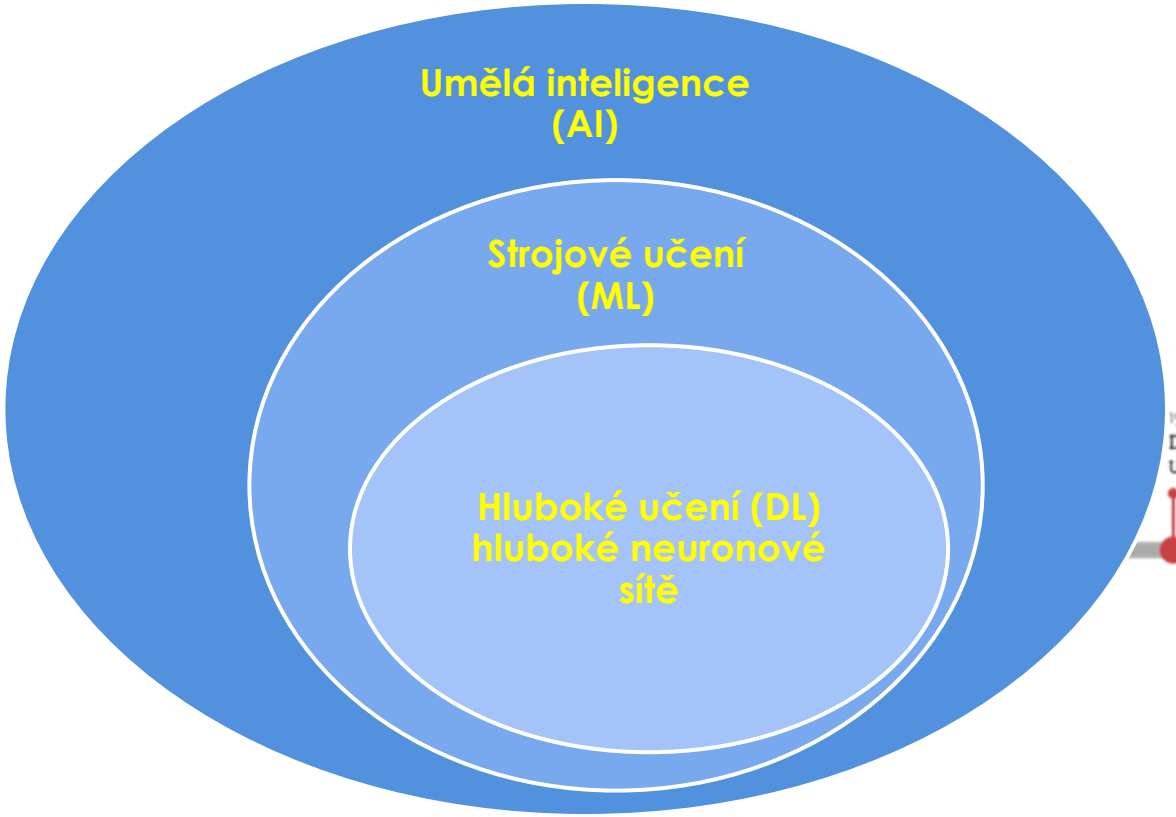
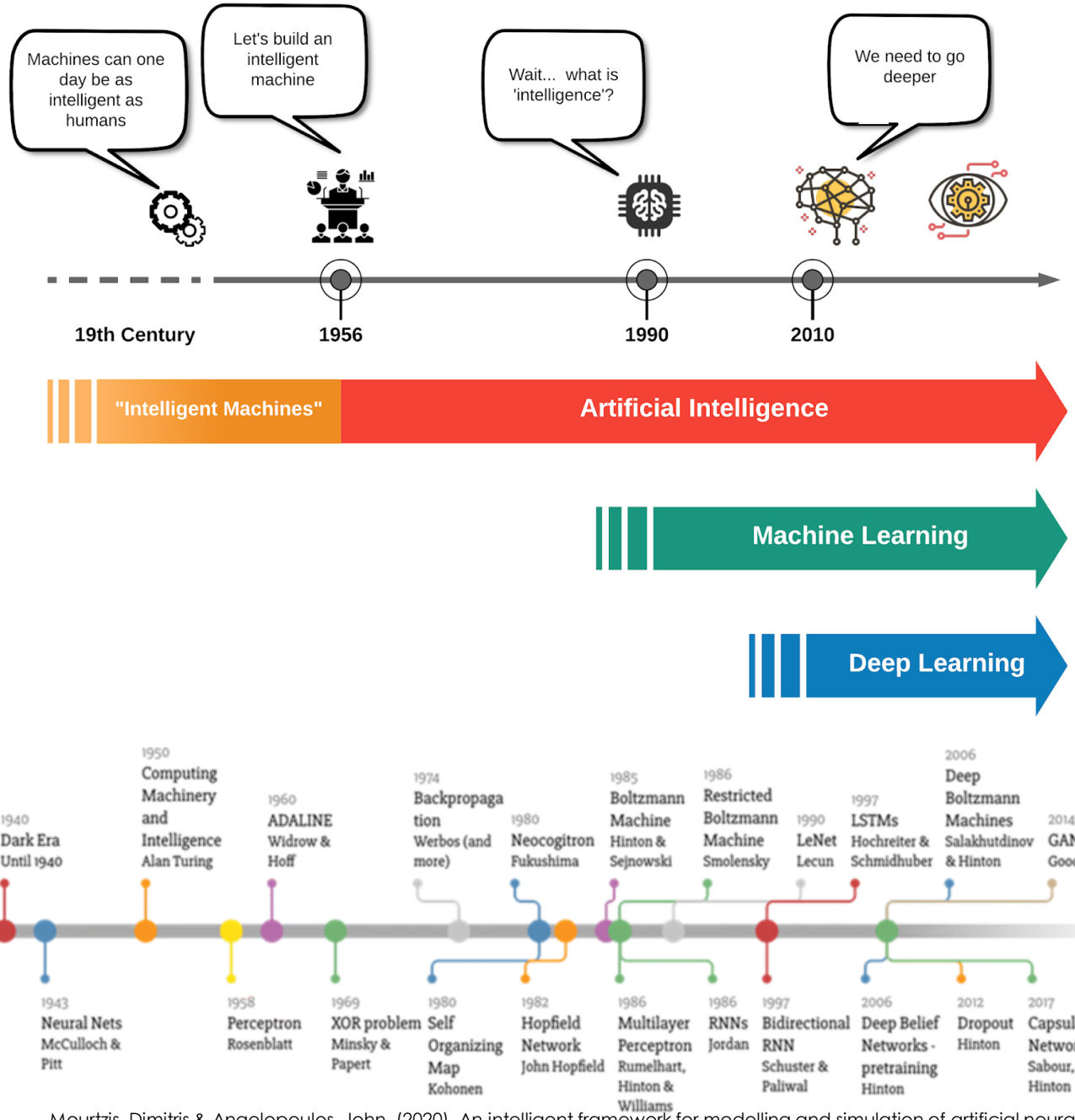
Hluboké učení a jeho aplikace

ZUZANA PETŘÍČKOVÁ
ČVUT V PRAZE, FJFI, KATEDRA KSI
SSSVT, 20. LISTOPADU 2024

Umělá inteligence a strojové učení



https://www.codesofinterest.com/p/build-deeper.html



Mourtzis, Dimitris & Angelopoulos, John. (2020). An intelligent framework for modelling and simulation of artificial neural networks (ANNs) based on augmented reality. International Journal of Advanced Manufacturing Technology. 111. 10.1007/s00170-020-06192-y.

Co je to umělá inteligence (AI)?

- **Klasická definice:** schopnost strojů / počítačových programů napodobovat lidské schopnosti, které považujeme za **inteligentní:**
  - schopnost uvažovat a řešit problémy, plánovat
  - schopnost přizpůsobit se novému prostředí, učit se
  - kreativita
- Moderní definice: vědecká disciplína, která se zabývá návrhem **sofistikovaných systémů** pro řešení **komplexních problémů**
  - rozpoznávání obrazu, jazykový překlad, hraní šachů, medicínská diagnostika, autonomní vozidla,…
  - S AI se setkáváme na každém kroku: personalizace obsahu na soc. sítích, personalizované reklamy, detekce spamu, vyhledávače,…

ČVUT
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Strojové učení

- Modely a techniky, které umožňují počítačovému systému **učit se na základě** dat nebo předchozích zkušeností
- Počítačový systém **se vytvoří sám** bez nutnosti explicitního programování
- Často inspirované biologickými principy:

Tři typy modelů strojového učení

Supervised learning (učení s učitelem)

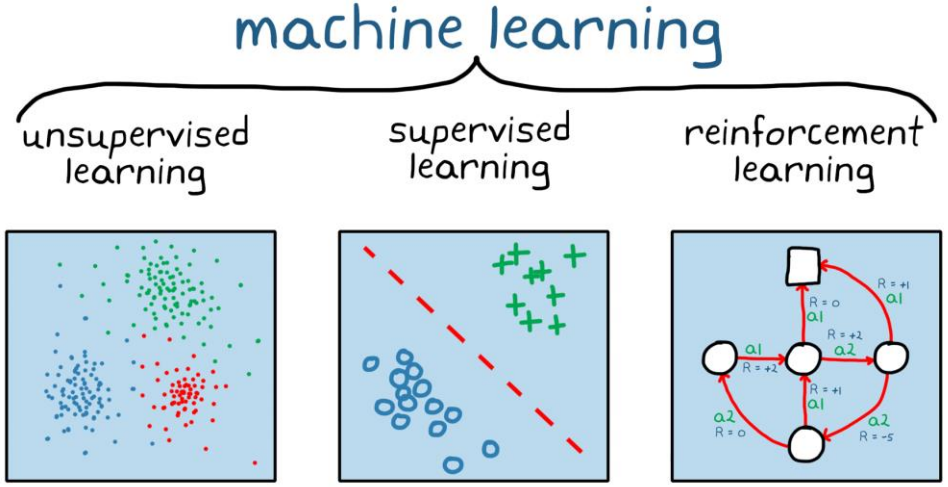
- učí se na označených datech (např. klasifikace obrázků)
- trénovací množina tvaru [vstup, požadovaný výstup]

Unsupervised learning (učení bez učitele)

- hledá strukturu v neoznačených datech (např. třídění obrázků)
- trénovací množina tvaru [vstup]
- př. detekce plagiátů či anomálií, recommendation systems

Reinforcement learning (Zpětnovazebné učení)

- učí se optimální strategii na základě zkušeností: prostřednictvím odměn a trestů v prostředí (např. robotický fotbal)
- př. herní průmysl, robotika, správa zdrojů, strojový překlad

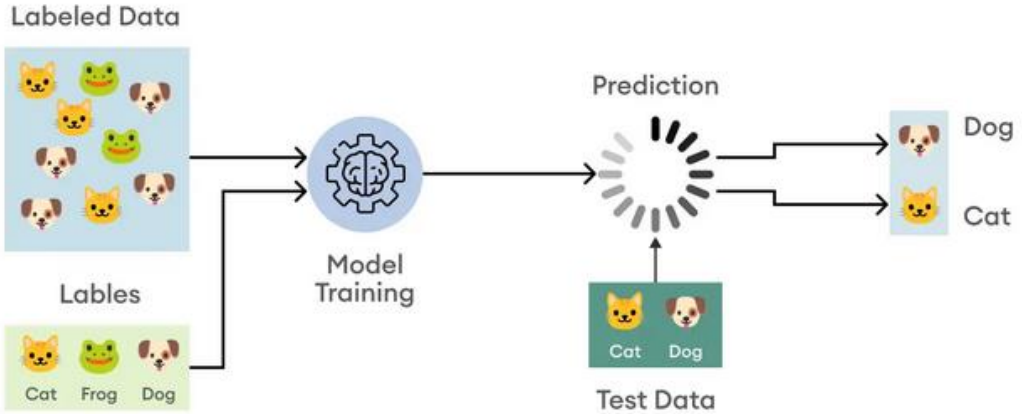


https://www.mathworks.com/discovery/reinforcement-learning.html

Učení s učitelem (supervised learning)

- **Trénovací množina** tvaru [vstup, požadovaný výstup]
- **Cíl učení:** aby model pro každý vstupní vzor správně predikoval hodnotu výstupu
- **Zobecňování:** model by měl dát správný výstup i pro data, která v trénovací množině nebyla
- **Typy úloh:** klasifikace, regrese, učení strukturovaných dat
- **Aplikace:** diagnostika v medicíně, klasifikace či segmentace obrazu, detekce fraudů (finanční podvody,…), rozpoznávání řeči, zpracování přirozeného jazyka,…

Klasifikace:



https://www.superannotate.com/blog/image-classification-basics

Regrese: **predikce numerické hodnoty (cena, teplota, sklon písma,…)**

Hluboké učení (deep learning)

- Využívá **umělé neuronové sítě** s mnoha vrstvami (tzv. hluboké sítě), které napodobují strukturu a funkci lidského mozku.
  - Model sám extrahuje příznaky z dat a snižuje nároky na jejich předzpracování
- **Výhody:**
  - Schopnost zpracovat obrovské množství dat a nalézt v nich složité vzory.
  - Flexibilita v různých typech úloh (klasifikace, rozpoznávání, generování obsahu,…).
- **Nevýhody:**
  - Vyžadují velké množství dat a výpočetní výkon.
  - Fungují jako „černá skříňka“.

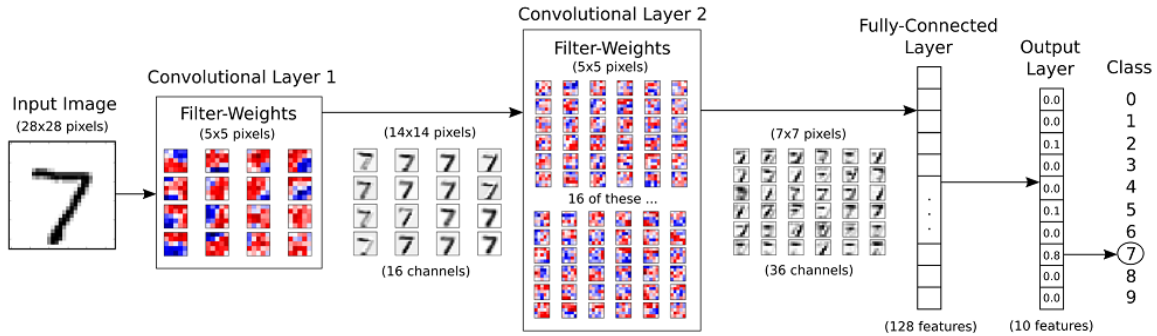
Klasické strojové učení:

Data → Extrakce příznaků → Aplikace modelu → Výstup

Hluboké učení:

Data → Aplikace modelu (model příznaky extrahuje sám) → Výstup

Příklad extrakce příznaků pomocí konvoluční neuronové sítě:

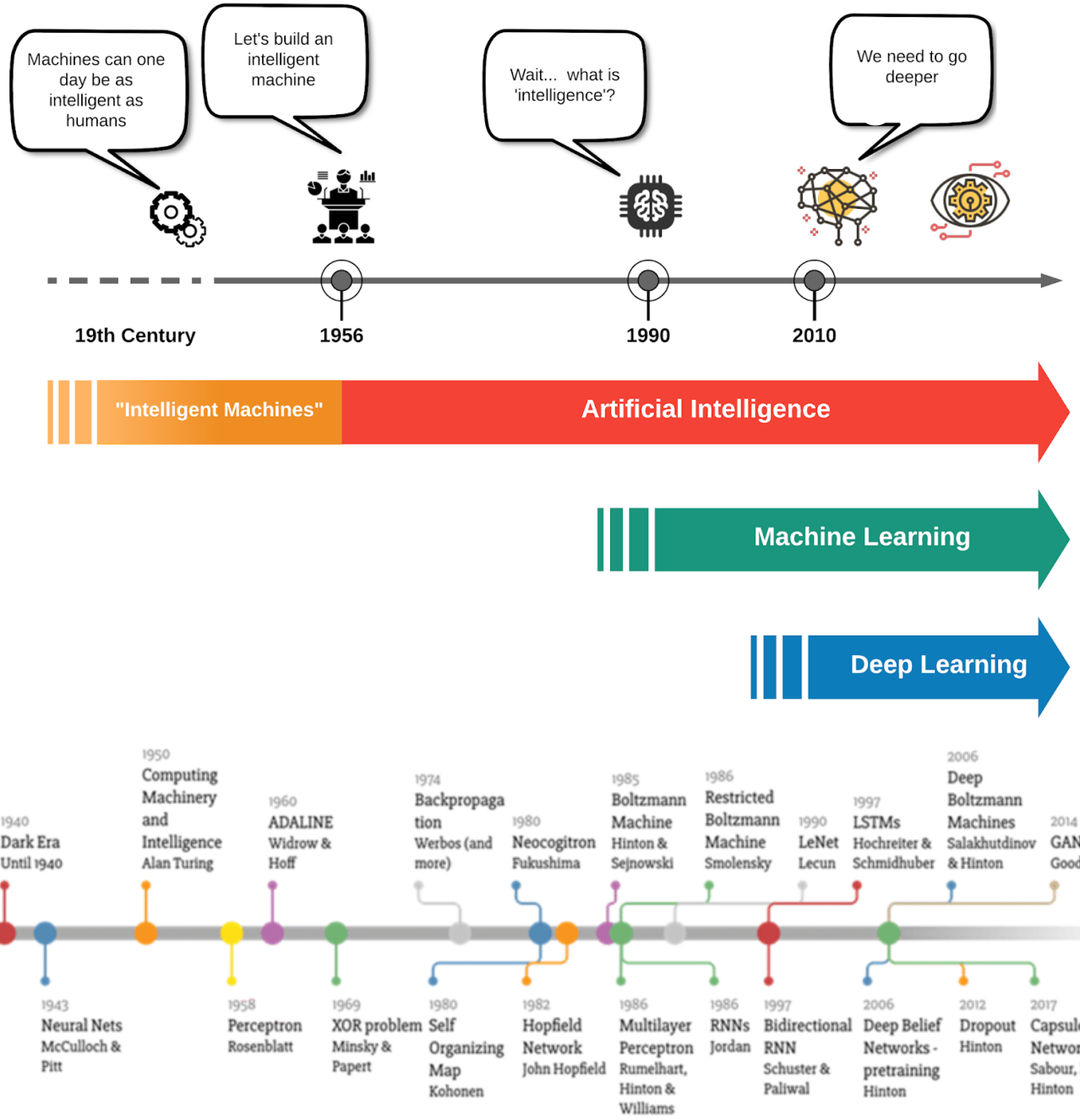


https://tensorflownet.readthedocs.io/en/latest/ConvolutionNeuralNetwork.html

Letmý pohled do historie AI

- **Počátky a raný vývoj (1940–1960)**
  - **První teoretické základy:** Turingův test (1950).
  - Vytvoření prvních programů pro hry a logiku.
- **Dvě "Neuronové zimy" (1970–1980, 2000–2010)**
  - Zklamání z nerealizovaných očekávání.
  - Klesající financování a zájem o AI výzkum.
- **Klíčové výzvy (1990–2012)**
  - **Výpočetní výkon:** Potřeba pokročilého hardware.
  - **Dostupnost dat:** Rozvoj internetu a big data.
  - **Algoritmické problémy:** Omezující techniky strojového učení.
- **Současný rozvoj a nástroje (2012–současnost)**
  - **Pokrok v architekturách:** Konvoluční a rekurentní neuronové sítě, transformery.
  - **Vylepšení hardware:** Grafické procesory a cloud computing.
  - **Nástroje pro AI:** TensorFlow, PyTorch, Keras,..
- **Současné výzvy a etické otázky**
  - **Zneužívání AI:** Falešné informace, ztráta soukromí a předpojatost.
  - **Regulace a odpovědnost:** Jak zajistit bezpečné a spravedlivé použití AI.

https://www.codesofinterest.com/p/build-deeper.html

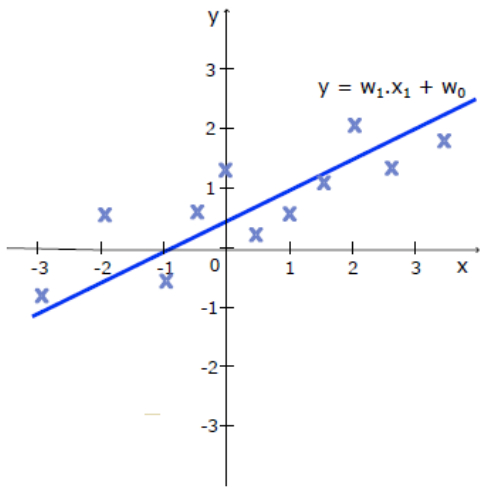


Mourtzis, Dimitris & Angelopoulos, John. (2020). An intelligent framework for modelling and simulation of artificial neural networks (ANNs) based on augmented reality. International Journal of Advanced Manufacturing Technology. 111. 10.1007/s00170-020-06192-y

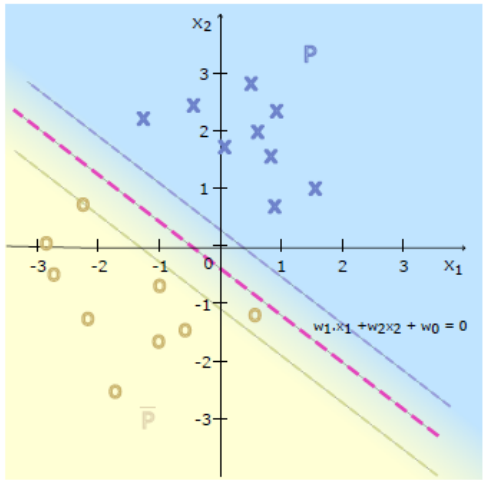
Jednoduchý neuron (perceptron)

- Původní, nejjednodušší model neuronové sítě (1957 – Frank Rosenblatt)
- Inspirován fungováním biologických neuronů
- Učení s učitelem (supervised learning)

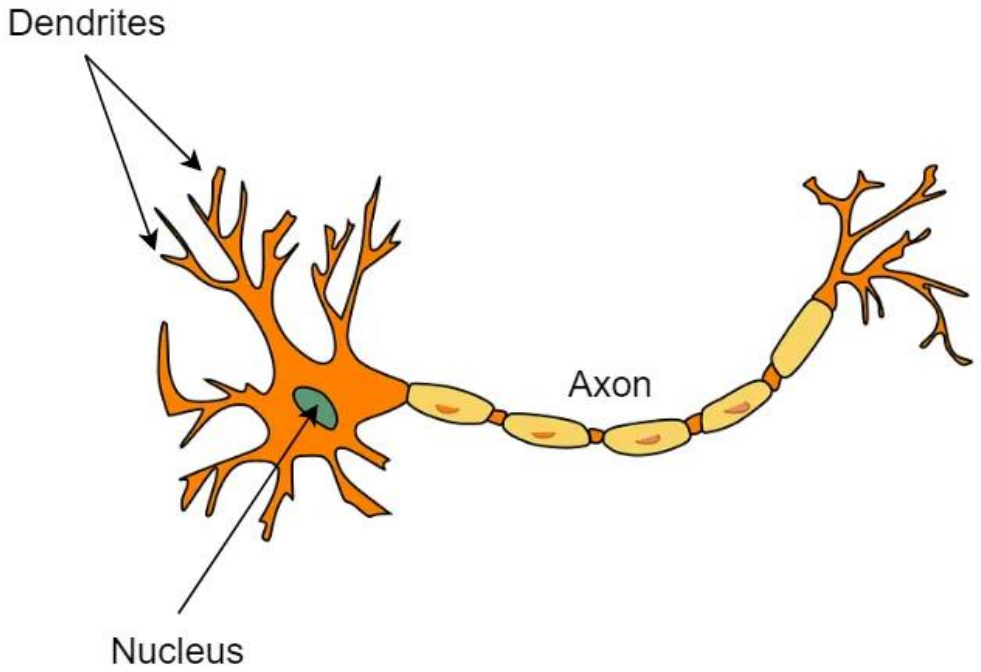
Lineární regrese:



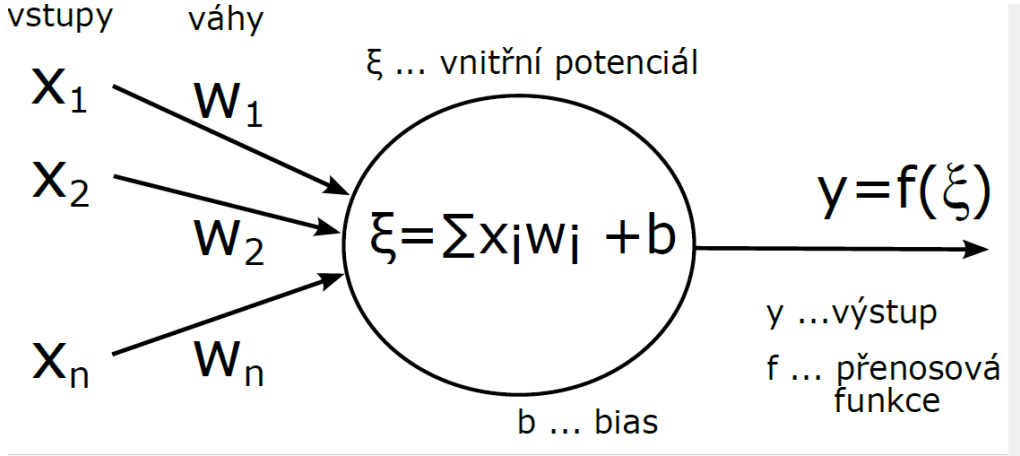
Binární klasifikace:



Biologický neuron:



Umělý neuron:



Jednoduchý neuron (perceptron) – ukázka implementace

C++:

```cpp
class Neuron {
private:
   vector<double> weights;
   double bias;
public:
   Neuron(int num_inputs) {
      for (int i = 0; i < num_inputs; i++) {
         weights.push_back((double) rand() / RAND_MAX);
      }
      bias = (double) rand() / RAND_MAX;
   }

   double feed_forward(vector<double> inputs) {
      double sum = 0;
      for (int i = 0; i < inputs.size(); i++) {
         sum += inputs[i] * weights[i];
      }
      sum += bias;
      return sigmoid(sum);
   }

   double sigmoid(double x) {
      return 1 / (1 + exp(-x));
   }
};
```

Python:

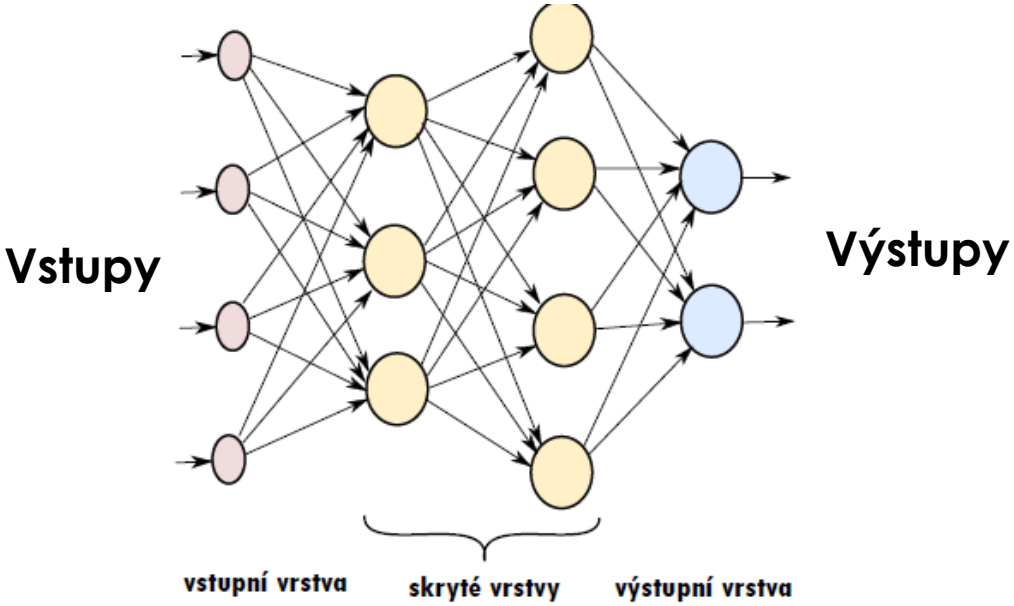
```python
import numpy as np

class Neuron:
   def __init__(self, num_inputs):
      self.weights = np.random.rand(num_inputs)
      self.bias = np.random.rand()

   def forward(self, inputs):
      net_input = np.dot(inputs, self.weights) + self.bias
      output = 1 / (1 + np.exp(-net_input))
      return output
```

Matematicky:

$$y = \sigma\left(w_0 + \sum_{k=1}^{n} w_k x_k\right)$$

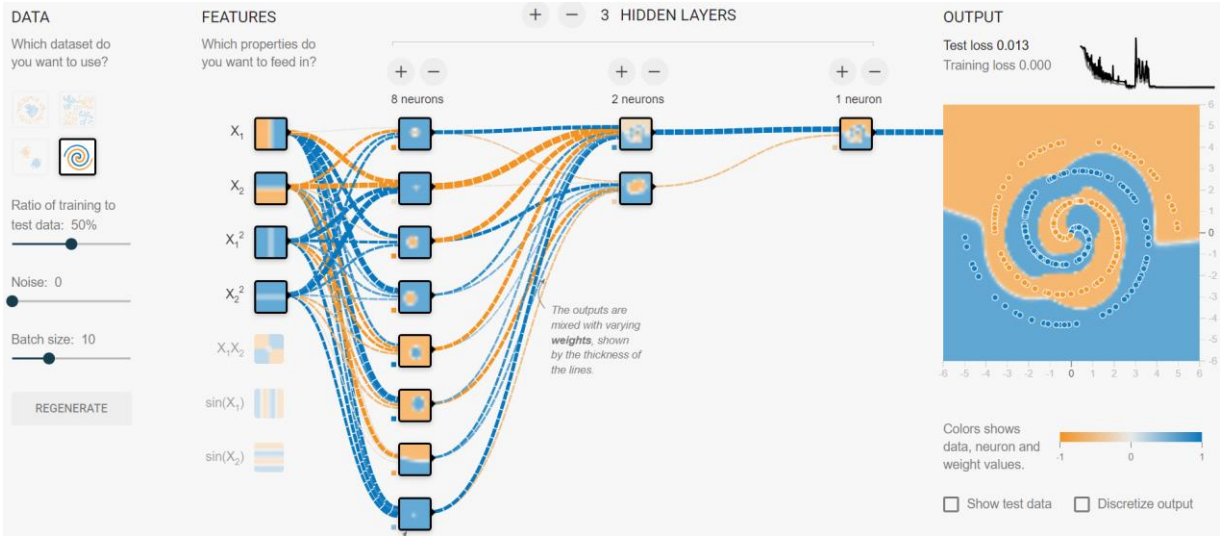
http://playground.tensorflow.org/

Vrstevnatá neuronová síť (Vícevrstvý perceptron, MLP)



- Klasický, univerzální model neuronové sítě
- Neurony jsou uspořádané do vrstev, vrstvy jsou plně propojené
- Vstupní vzory musí být vektory čísel
- Učení gradientní metodou pomocí algoritmu zpětného šíření (backpropagation)
  - **Cíl učení:** nastavit parametry modelu (váhy a biasy všech neuronů) tak, aby model pro každý vstupní vzor dal správný výstup

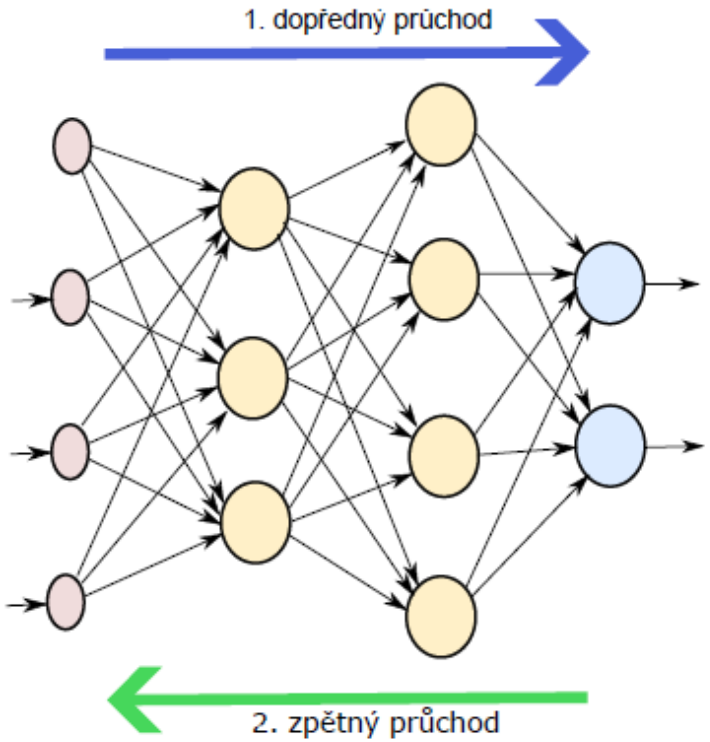
http://playground.tensorflow.org/



Učení neuronových sítí – algoritmus zpětného šíření (backpropagation)

Základní princip učení neuronových sítí:

- Náhodně inicializujeme parametry modelu (váhy a prahy)
- V cyklu:
  - Připravíme si dávku trénovacích vzorů
  - Spočteme skutečný výstup (**dopředný průchod**)
  - Spočítáme chybu modelu: jak moc se liší skutečný výstup modelu od požadovaného výstupu
  - Adaptujeme váhy a prahy (**zpětný průchod**): tak, aby se chyba modelu o něco zmenšila



Vrstevnatá neuronová síť (MLP) – ukázka implementace

C++:

```cpp
class MLP {
public:
    MLP(int input_size, int hidden_size, int output_size) {
        input_hidden_weights = Eigen::MatrixXd::Random(input_size, hidden_size);
        hidden_output_weights = Eigen::MatrixXd::Random(hidden_size, output_size);
        hidden_bias = Eigen::VectorXd::Random(hidden_size);
        output_bias = Eigen::VectorXd::Random(output_size);
    }

    Eigen::VectorXd forward(const Eigen::VectorXd& inputs) {
        Eigen::VectorXd hidden_output = sigmoid((inputs.transpose() * input_hidden_weights).transpose() + hidden_bias);
        Eigen::VectorXd output = sigmoid((hidden_output.transpose() * hidden_output_weights).transpose() + output_bias);
        return output;
    }

private:
    double sigmoid(double x) {
        return 1 / (1 + exp(-x));
    }
    Eigen::VectorXd sigmoid(const Eigen::VectorXd& x) {
        return 1 / (1 + (-x.array()).exp());
    }

    Eigen::MatrixXd input_hidden_weights;
    Eigen::MatrixXd hidden_output_weights;
    Eigen::VectorXd hidden_bias;
    Eigen::VectorXd output_bias;
};
```

Python:

```python
import numpy as np

class MLP:
    def __init__(self, input_size, hidden_size, output_size):
        self.input_hidden_weights = np.random.rand(input_size, hidden_size)
        self.hidden_output_weights = np.random.rand(hidden_size, output_size)
        self.hidden_bias = np.random.rand(hidden_size)
        self.output_bias = np.random.rand(output_size)

    def forward(self, inputs):
        hidden_output = self.sigmoid(np.dot(inputs, self.input_hidden_weights) + self.hidden_bias)
        output = self.sigmoid(np.dot(hidden_output, self.hidden_output_weights) + self.output_bias)
        return output

    def sigmoid(self, x):
        return 1 / (1 + np.exp(-x))
```

Matematicky:

$$\mathbf{h} = \varphi(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

$$\mathbf{y} = \sigma(\mathbf{W}^{(2)}\mathbf{h} + \mathbf{b}^{(2)})$$

http://playground.tensorflow.org/

Knihovny a frameworky pro hluboké učení

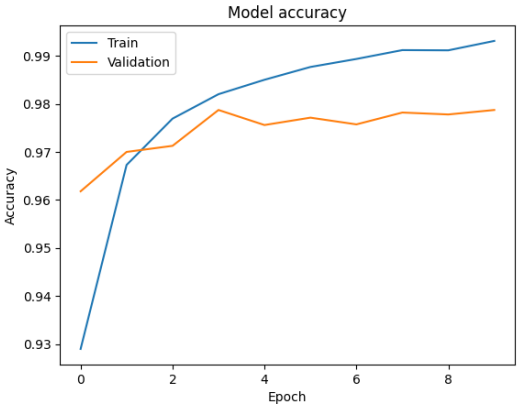
- Řada frameworků a dalších nástrojů:
  - **Python:**
    - **TensorFlow (2015)**
    - **PyTorch (2016)**
    - **Keras (2015)**
    - Caffe (2013)
    - Theano (2007)
    - MXNet (2015)
    - …
  - **Java:**
    - Deeplearning4j (2014)
    - Neuroph (2008)
    - DL4J (2014)
  - **C++:**
    - Caffe (2013)
    - Torch (2002)
    - TensorFlow C++ (2015)
  - **R:**
    - MXNet (2015)
    - TensorFlow (2015)
    - Keras (2015)
    - H2O.ai (2012)
  - **Julia:**
    - Flux.jl (2016)
    - Knet.jl (2017)

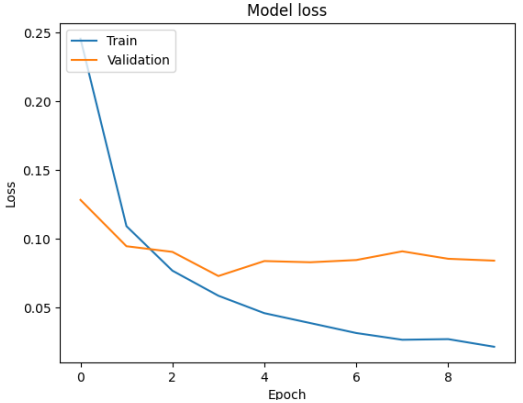
- Dostupnost dat k učení (např. https://www.kaggle.com/datasets)
- Řada předučených modelů (např. https://keras.io/api/applications/)
- Velké množství praktických příkladů (use cases, https://www.kaggle.com/code)

Vrstevnatá neuronová síť (MLP) – Keras

```
import keras
from keras.datasets import mnist
import numpy as np
from sklearn.model_selection import train_test_split

################################################
# Load  and pre-proces the data
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(60000, 28 * 28)
x_test = x_test.reshape(10000, 28 * 28)
x_train = x_train.astype('float32') / 255
x_test = x_test.astype('float32') / 255
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train)

################################################
# Define the model
# Define architecture:
model = keras.Sequential([
    keras.layers.InputLayer(shape=(28 * 28,)),
    keras.layers.Dense(512, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(10, activation='softmax')
])
# Summarize the model
model.summary()
```

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

################################################
# Train the model
history = model.fit(x_train, y_train, epochs=8, , batch_size=64,
                    validation_data=(x_val, y_val))
################################################
# Evaluate the model on the training set
train_loss, train_acc = model.evaluate(x_train, y_train)
print('Training accuracy:', train_acc, '\nTrain loss:', train_loss)

# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test accuracy:', test_acc, '\nTest loss:', test_loss)

################################################
# Use the model to make predictions
y_pred_probs = model.predict(x_test)  # get probabilities
y_pred = np.argmax(y_pred_probs, axis=1) # get predicted labels
print("Predicted labels:", y_pred[:10], "\nTrue labels:", y_test[:10])
```

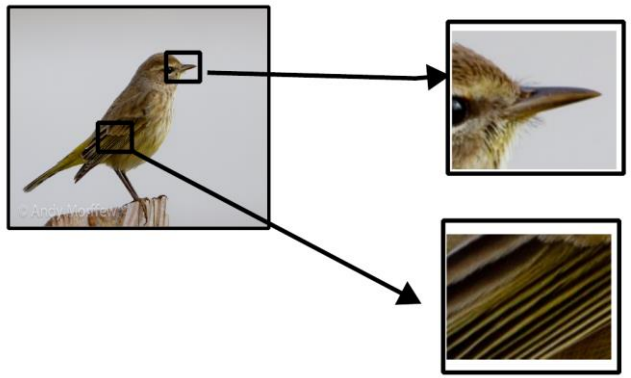




Open code in github
https://github.com/reitezuz/notebooks-for-NES2-2024

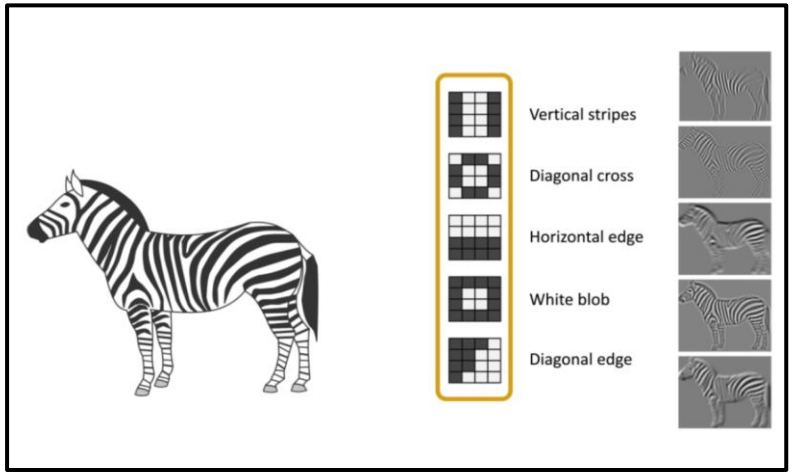
Konvoluční neuronové sítě (CNNs)

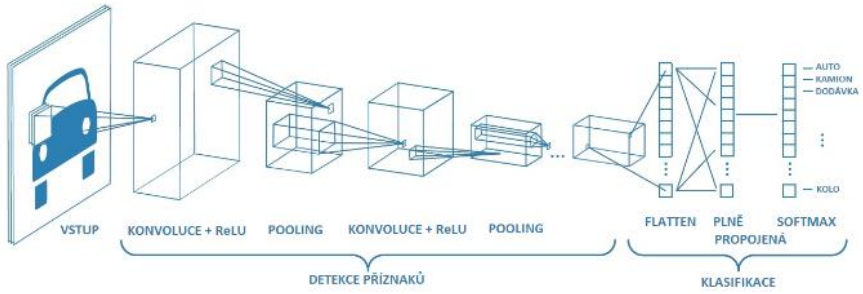
- Založeny na operaci konvoluce
- Jsou schopné rozpoznat v datech vzory (příznaky) různé úrovně: od hran až např. po oči



- **Hlavní aplikace:** zpracování obrazu (klasifikační úlohy, rozpoznání a segmentace obrazu), zpracování signálů, videa

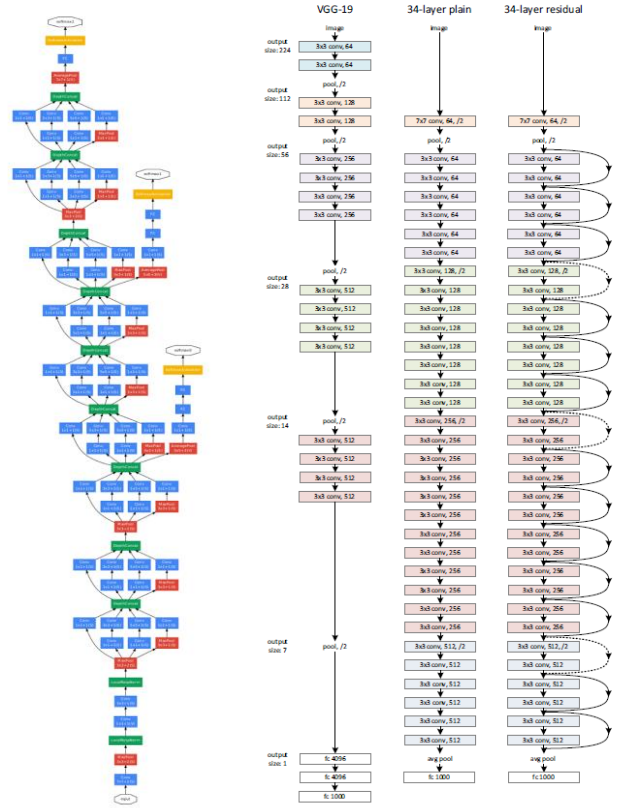
Ukázka příznaků extrahovaných první konvoluční vrstvou:





Zdroj : https://matlabacademy.mathworks.com/details/deep-learning-onramp/deeplearning, upraveno

GoogleNet **ResNet**

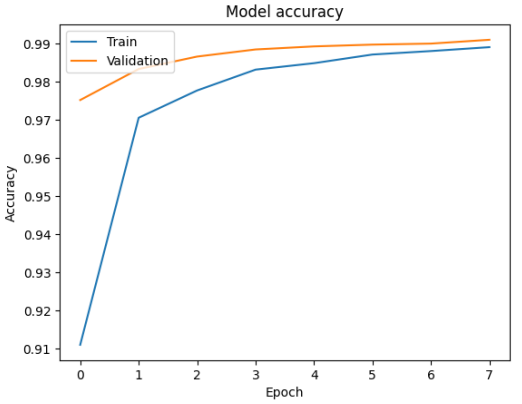


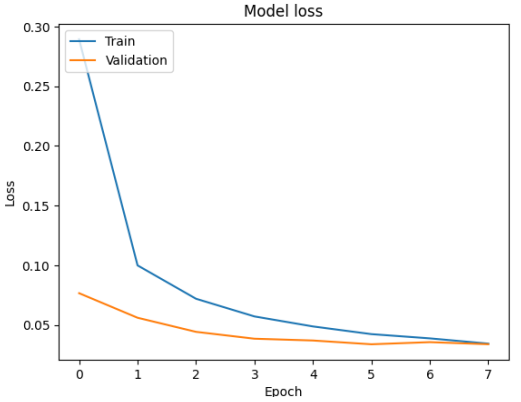
Konvoluční neuronové sítě (CNN) – Keras

```
import keras
from keras.datasets import mnist
import numpy as np
from sklearn.model_selection import train_test_split

################################################
# Load  and pre-proces the data
(train_images, y_train), (test_images, y_test) = mnist.load_data()
x_train = train_images.reshape(-1, 28, 28, 1).astype('float32') / 255
x_test = test_images.reshape(-1, 28, 28, 1).astype('float32') / 255 x_train,
x_val, y_train, y_val = train_test_split(x_train, y_train)

################################################
# Define the model architecture
model = keras.Sequential([
    keras.Input(shape=(28, 28, 1)),
    keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu'),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Conv2D(64, kernel_size=(3, 3), activation='relu'),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(10, activation='softmax')
])
 # Summarize the model
model.summary()
```

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

################################################
# Train the model
history = model.fit(x_train, y_train, epochs=8, batch_size=64,
                    validation_data=(x_val, y_val))
################################################
# Evaluate the model on the training set
train_loss, train_acc = model.evaluate(x_train, y_train)
print('Training accuracy:', train_acc, '\nTrain loss:', train_loss)

# Evaluate the model on the test set
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test accuracy:', test_acc, '\nTest loss:', test_loss)

################################################
# Use the model to make predictions
y_pred_probs = model.predict(x_test)  # get probabilities
y_pred = np.argmax(y_pred_probs, axis=1) # get predicted labels
print("Predicted labels:", y_pred[:10], "\nTrue labels:", y_test[:10])
```

Open code in github
https://github.com/reitezuz/notebooks-for-NES2-2024





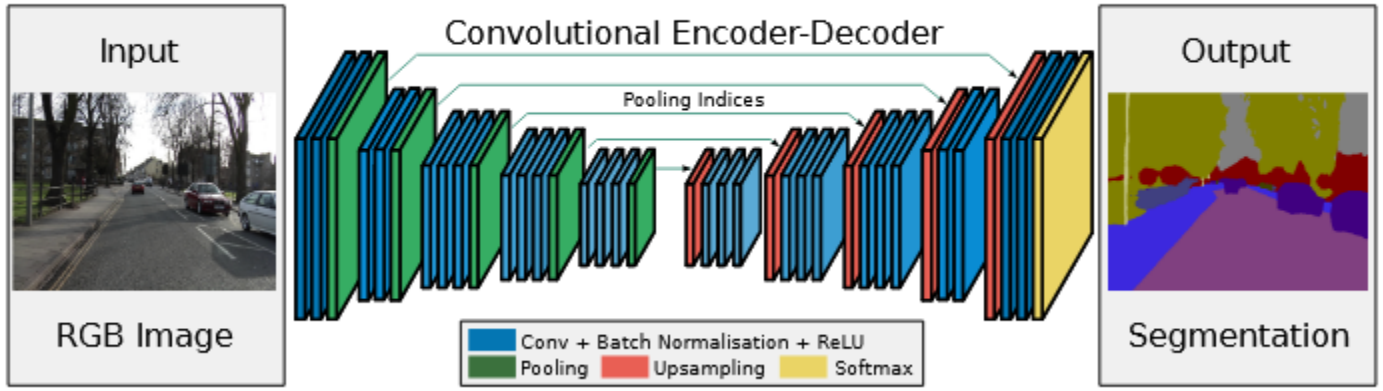
Architektury hlubokých neuronových sítí

Autoencodery

- Model hluboké neuronové sítě učený bez učitele
- Změna reprezentace dat, čištění dat, komprese a rekonstrukce

Encoder-decodery

- Segmentace obrazu, popis obrázků, shrnutí textů,…

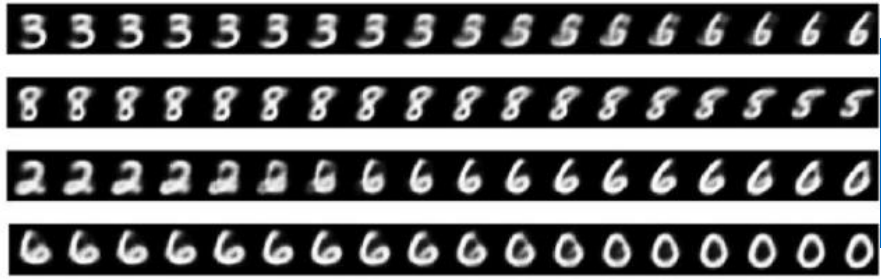


Zdroj : SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentationhttps://arxiv.org/pdf/1511.00561

Variační autoencodery:

- Kódují data jako pravděpodobnostní rozdělení
- Schopné generovat nová data nebo vytvářet interpolace:

Tom White: Sampling generative networks, https://arxiv.org/pdf/1609.04468

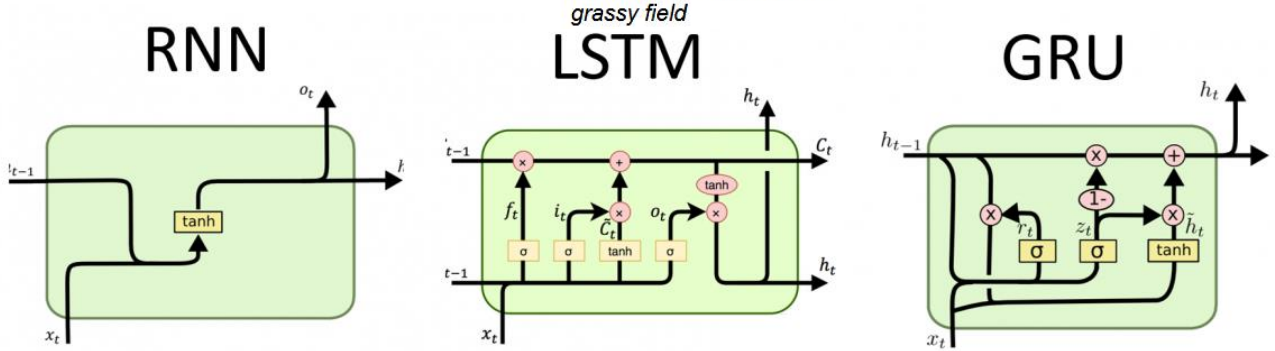


F. Chollet: Deep learning v jazyku Python, obr. 5.7

Architektury hlubokých neuronových sítí

- **Rekurentní neuronové sítě (RNNs)**
  - Analýza sekvenčních dat (časové řady, řeč, text, písmo)
- **Sítě s dlouhou-krátkodobou pamětí (Long Short-Term Memory Networks, LSTMs)**
  - Zpracování jednorozměrných signálů a časových řad (např. rozpoznání řeči a písma)
  - Model schopný zachytit i dlouhodobé závislosti
- **Gated Recurrent Unit Networks (GRU)**
  - Zjednodušená verze LSTM
  - Modelování sekvenčních dat
  - Rozpoznání řeči a strojový překlad

Příklad: Image captioning (popis obrázků):

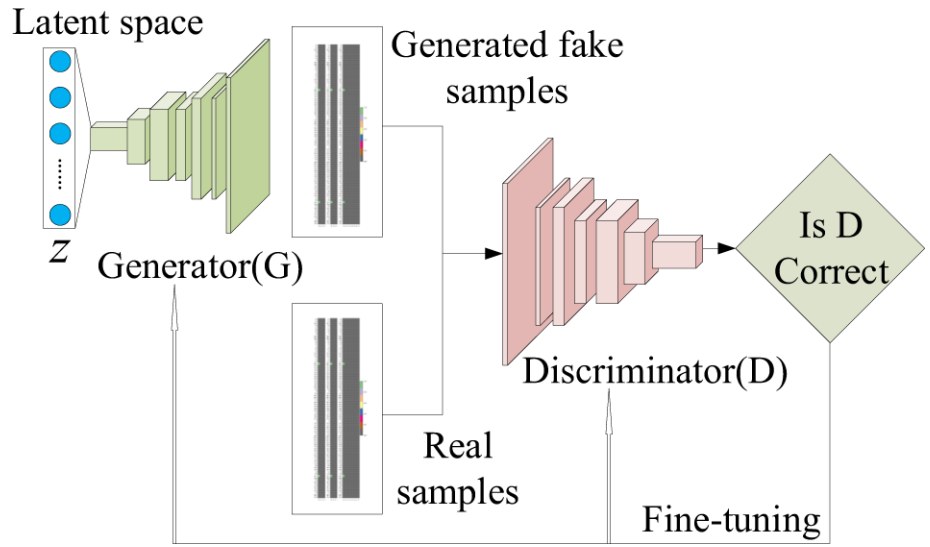


https://cs231n.stanford.edu/slides/2020/lecture_10.pdf

Architektury hlubokých neuronových sítí

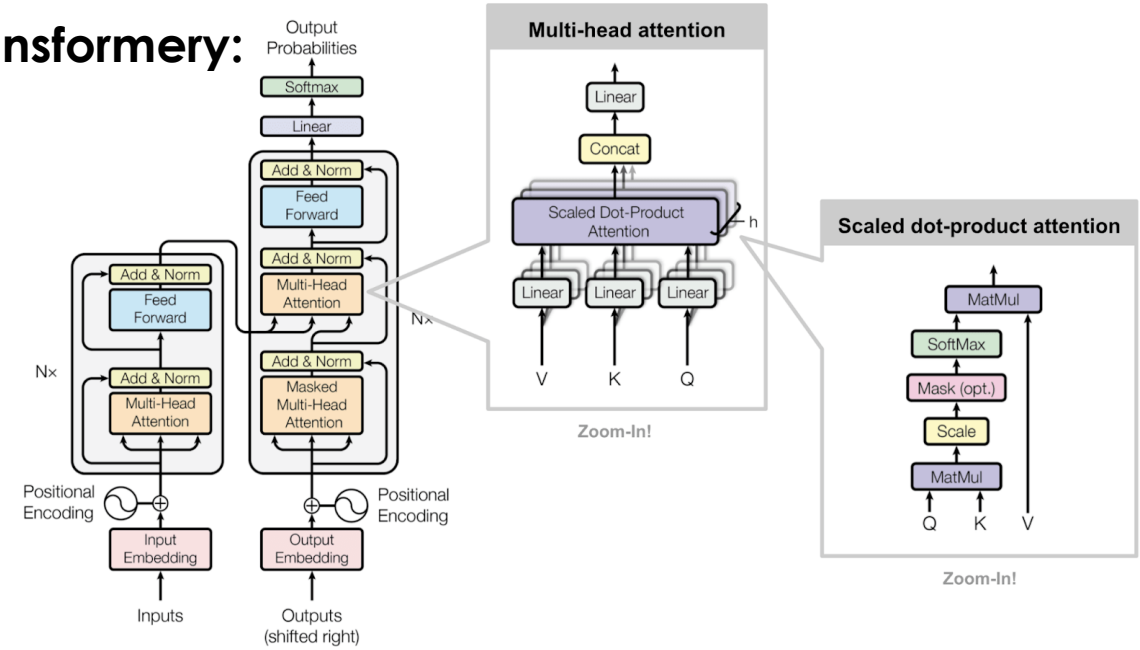
- **Generative Adversarial Networks (GANs)**
  - Generování nových dat na základě naučených vzorů.
- **Siamese Networks**
  - Pro úlohy rozpoznání obrazu, object-tracking
  - Počítají podobnost mezi dvěma různými vstupy
- **Capsule Networks**
  - Pro úlohy rozpoznání obrazu
  - Modelují hierarchické vztahy mezi částmi objektů
- **Transformer Networks (Gemini, GPT)**
  - Pro úlohy zpracování přirozeného jazyka (klasifikace textů, překlady, generování textu apod.).

Generativní modely:



Dan, Y., Zhao, Y., Li, X. *et al.* Generative adversarial networks (GAN) based efficient sampling of chemical composition space for inverse design of inorganic materials. *npj Comput Mater* **6**, 84 (2020). https://doi.org/10.1038/s41524-020-00352-0

Transformery:



Attention is all you need, A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, et.al.. Advances in Neural Information Processing Systems , page 5998--6008. (2017)

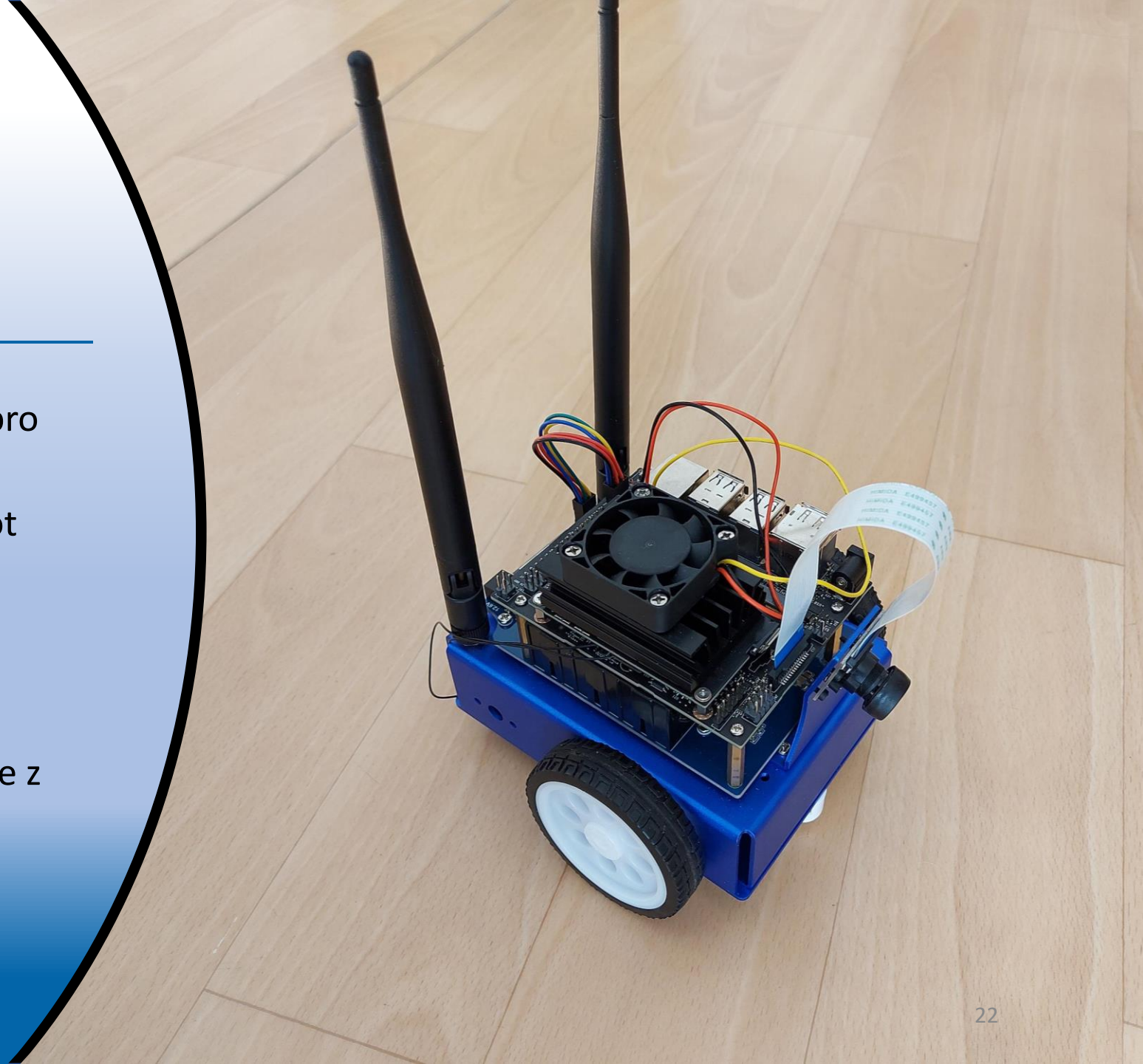
Hluboké učení v závěrečných pracích našich studentů

- Hlasově ovládaný robot
- Generování map pomocí generativního modelu neuronové sítě
- Analýza a zpracování obrazu (identifikace objektů a odstranění šumu,…)
- Autonomní agenti v počítačových hrách
- Fyzikální aplikace, vývoj pro CERN
- a mnohé další

Hlasem ovládaný robot

- Využití metod hlubokého učení (LSTM) pro rozpoznání řeči
- Moderní hardware Nvidia Jetson a Jetbot
- Možnost řízení přes wi-fi

- Online rozpoznávání hlasových příkazů
- Online rozpoznávání předmětů na obraze z kamery



Procedurální generování prostředí

- Generování map z náhodného šumu pomocí generického modelu neuronové sítě (GAN)
- Model učený na satelitních snímcích Země
- Využití v herním průmyslu

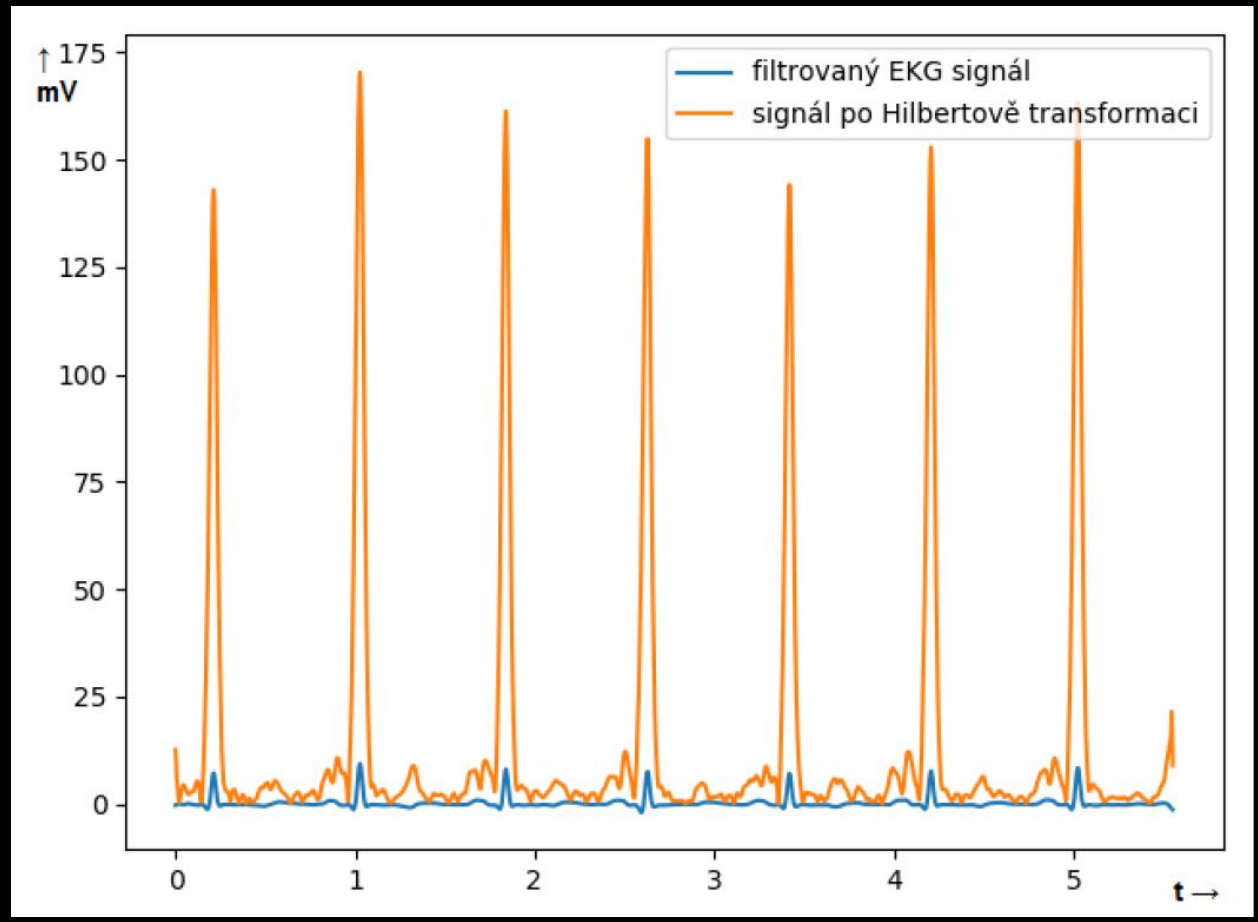
Obrázek 5.4: Ukázka zvolené mapy o velikosti 1000x1000 pixelů

Obrázek 6.3: Ukázka natrénované a negenerované sítě pomocí náhodného šumu

Aplikace strojového učení při zpracování medicínských dat

Klasifikace EKG signálů pomocí support vector machines v Pythonu

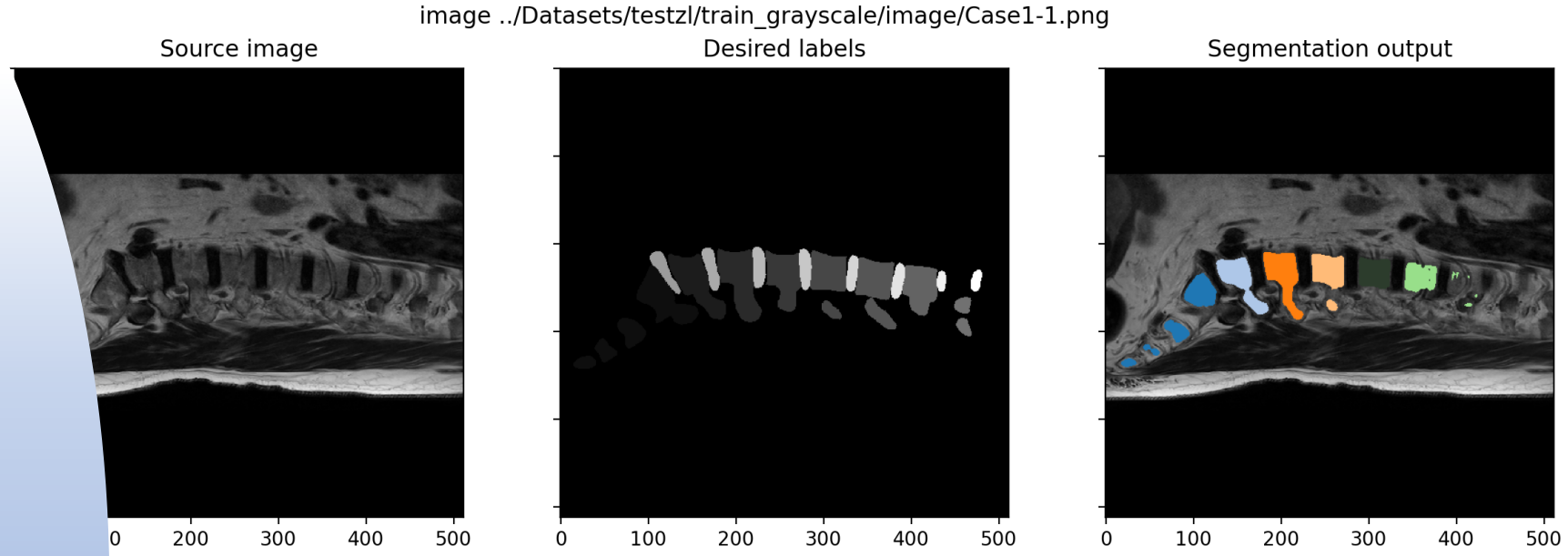
- Využití volně dostupných dat ručně anotovaných lékaři
- Předzpracování signálu
- Aplikace strojového učení na medicínský problém
- Úspěšnost 99 %

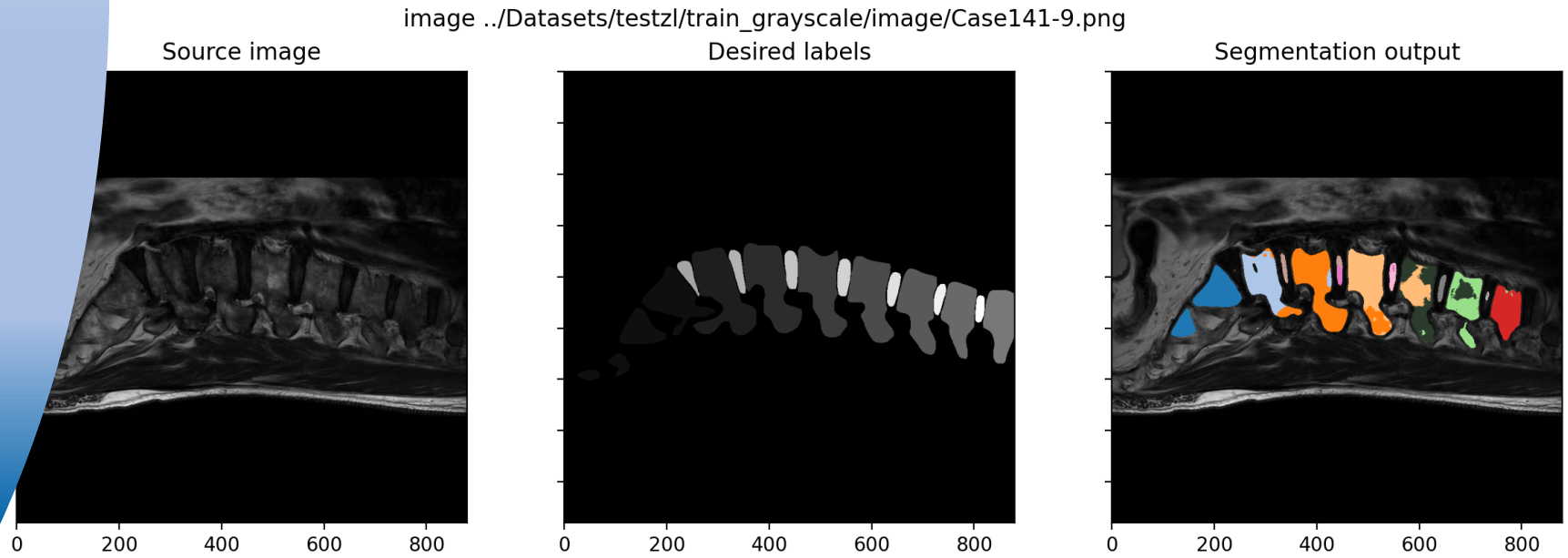


Aplikace strojového učení při zpracování medicinských dat

- Snímky RTG/MR/CT
- Podpora při tvorbě diagnózy

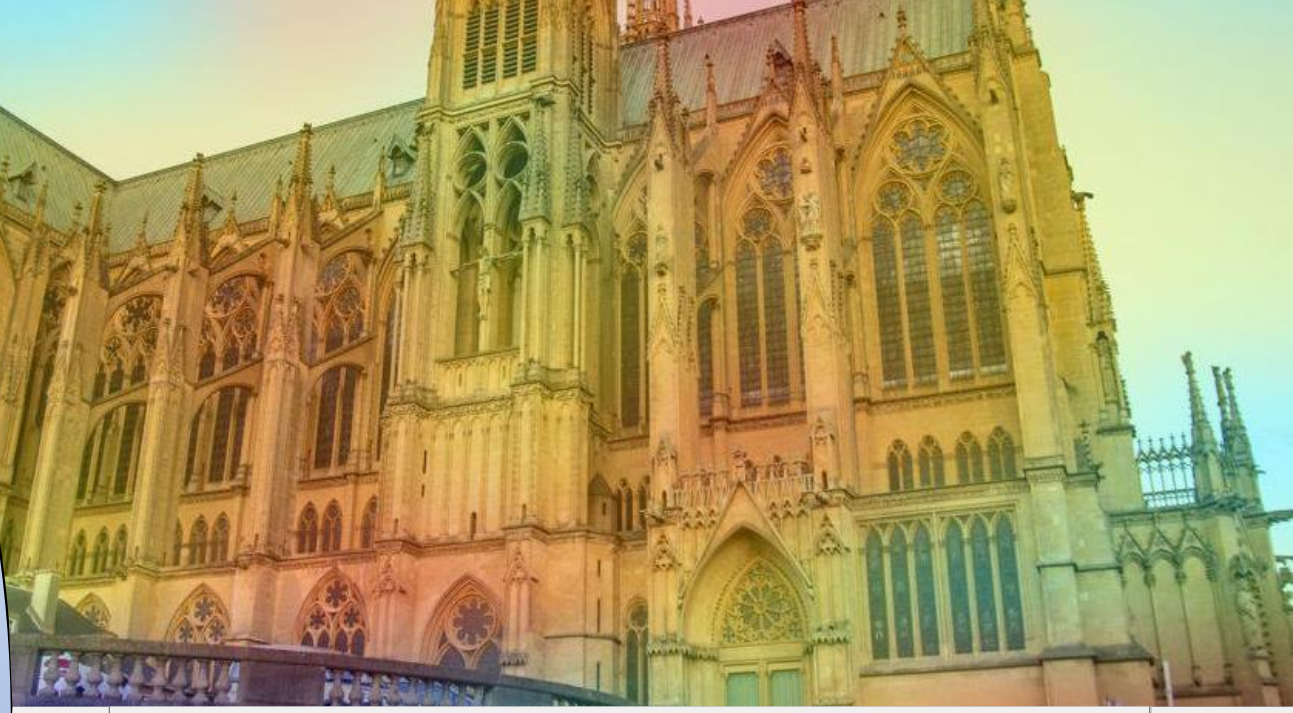
- Čištění obrazu, odstranění šumu
- Automatická segmentace, hledání objektů/struktur
- Klasifikace a predikce nalezených struktur

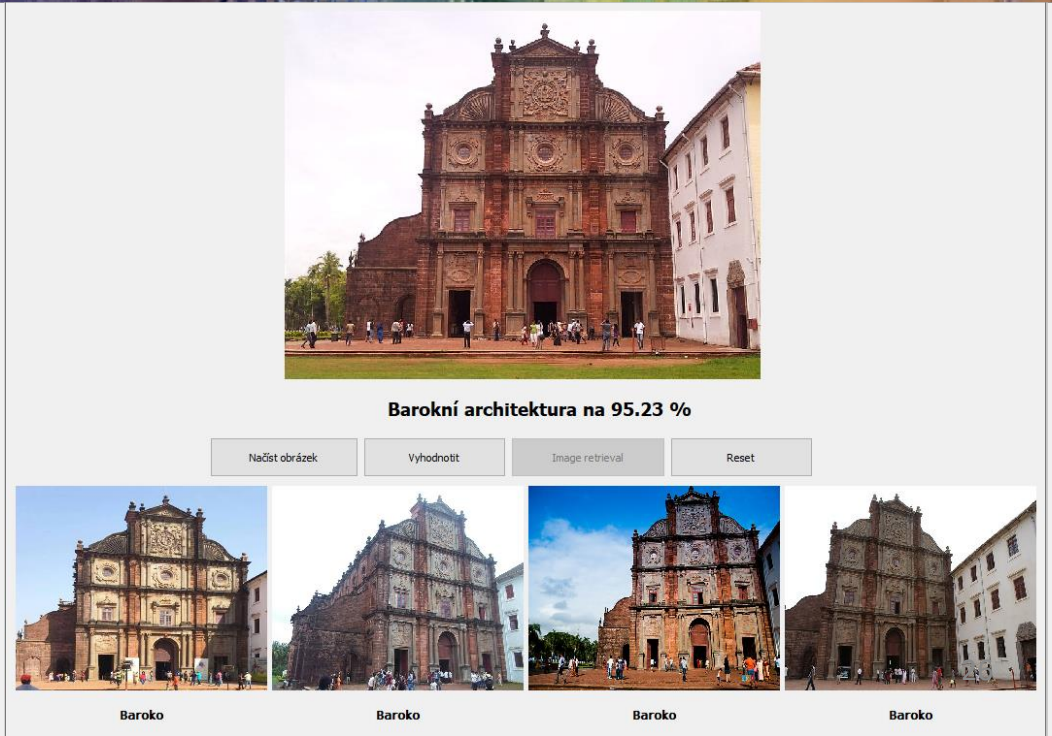




Využití konvolučních neuronových sítí pro rozpoznání obrazu

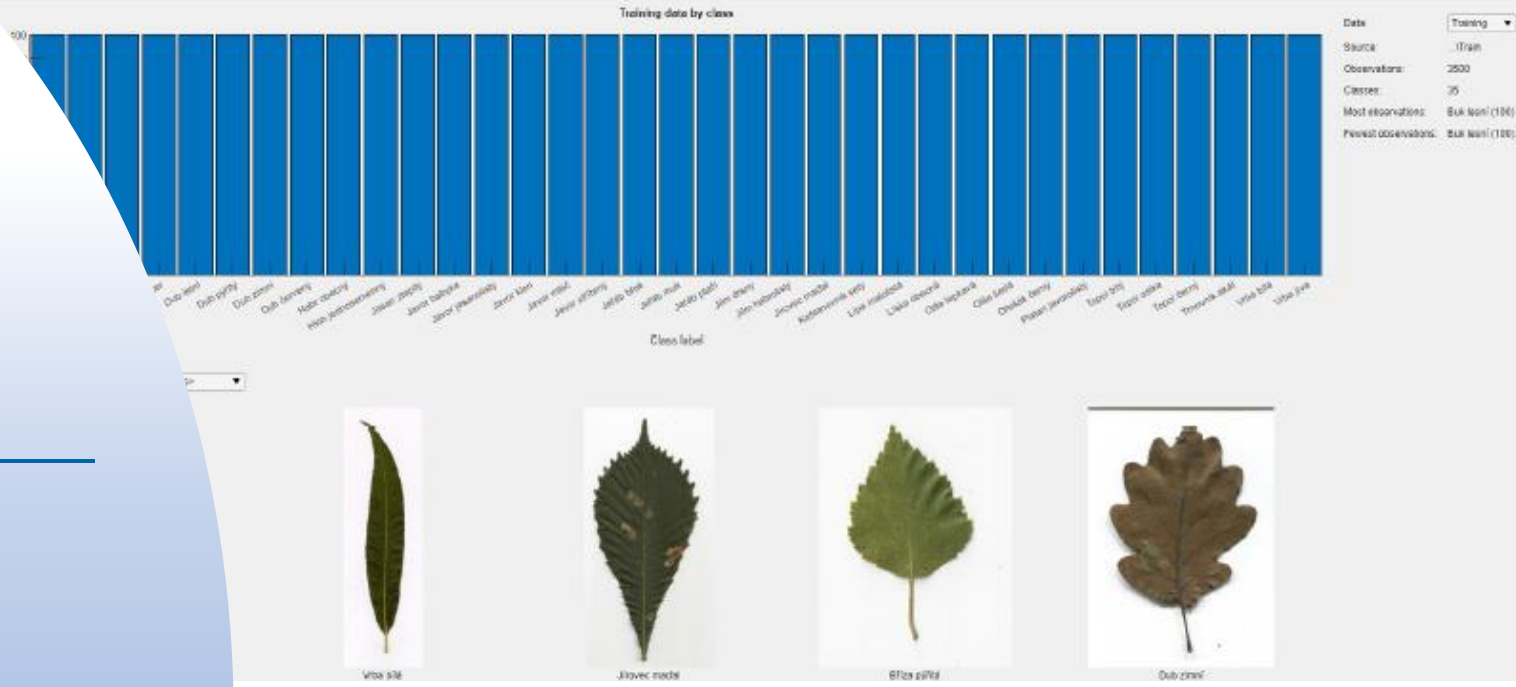
- Aplikace konvolučních neuronových sítí s využitím metody přeneseného učení
- Rozpoznávání architektonického stylu budovy z fotografie
- Vyhledání podobných fotografií v databázi
- Vizualizace zajímavých atributů naučené konvoluční sítě

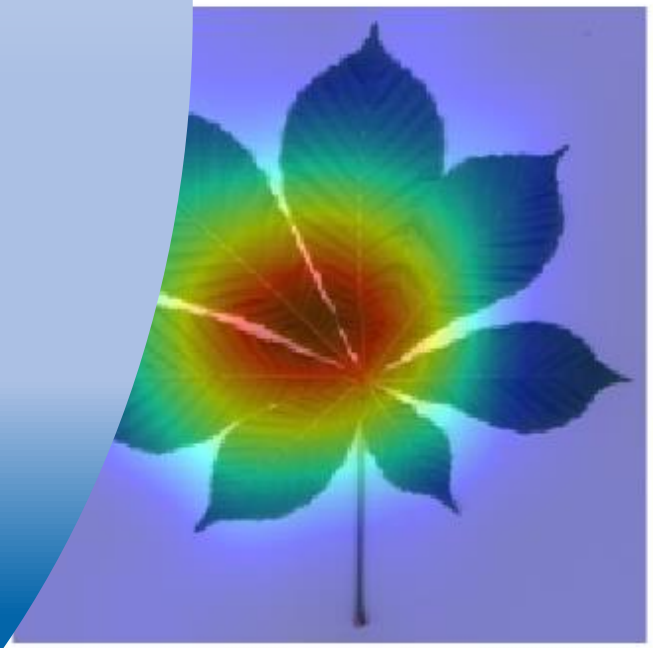




Využití konvolučních neuronových sítí pro rozpoznání obrazu

- Aplikace konvolučních neuronových sítí s využitím metody přeneseného učení
- Deep Network Designer (MATLAB)
- Vizualizace zajímavých atributů naučené konvoluční sítě
- Rozpoznávání listů stromů





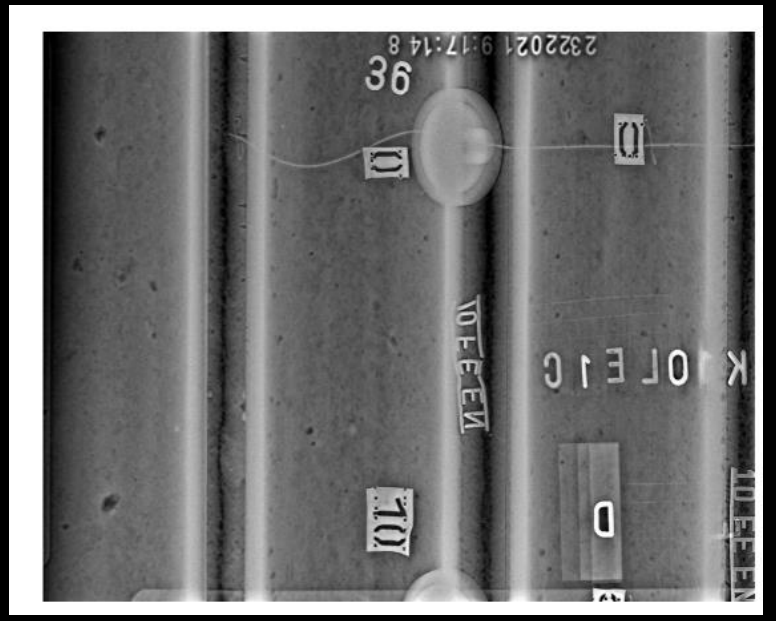
(a) Jírovec maďal

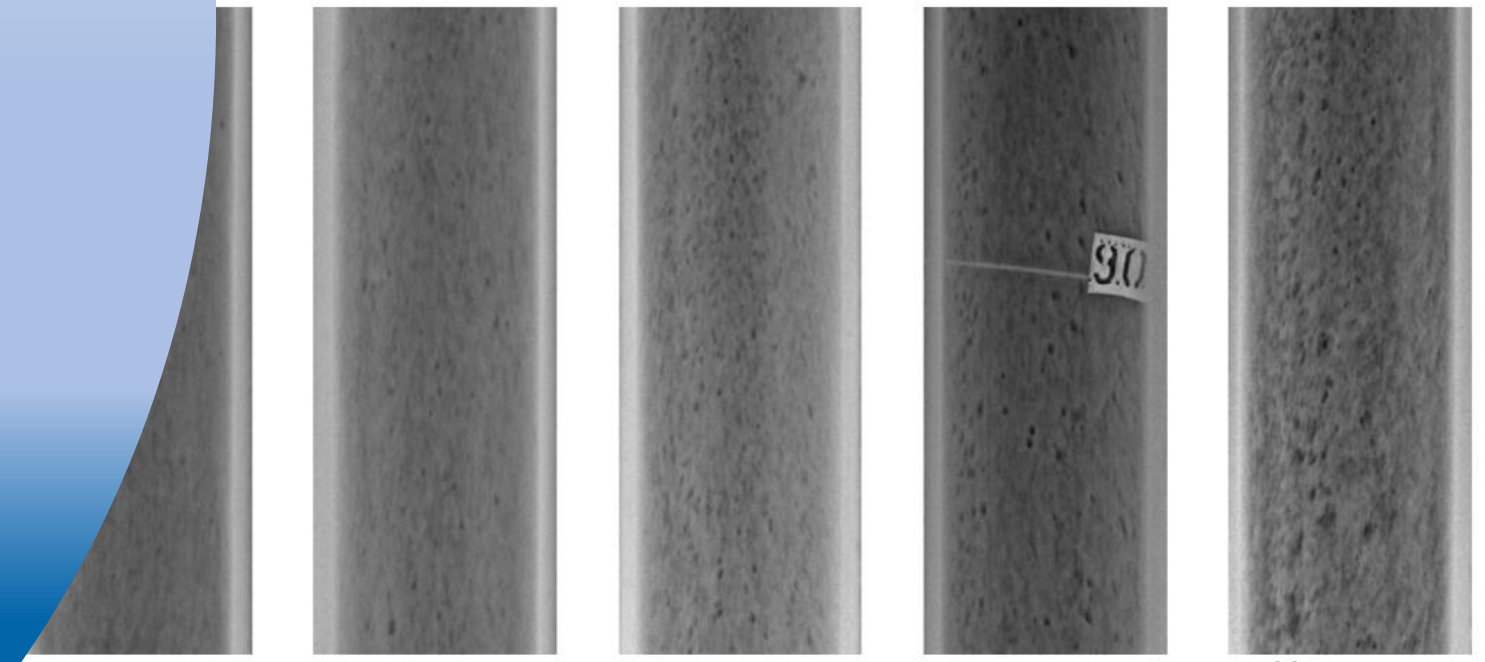
(b) Javor babyka

Využití konvolučních neuronových sítí pro rozpoznání obrazu

Klasifikace RTG snímků trubek kotle

- Detekce stupně důlkové koroze (třídy poškození Q1,…,Q5)
- Srovnání různých modelů CNN





Využití konvolučních neuronových sítí pro rozpoznání obrazu

Rozpoznávání typu automobilu

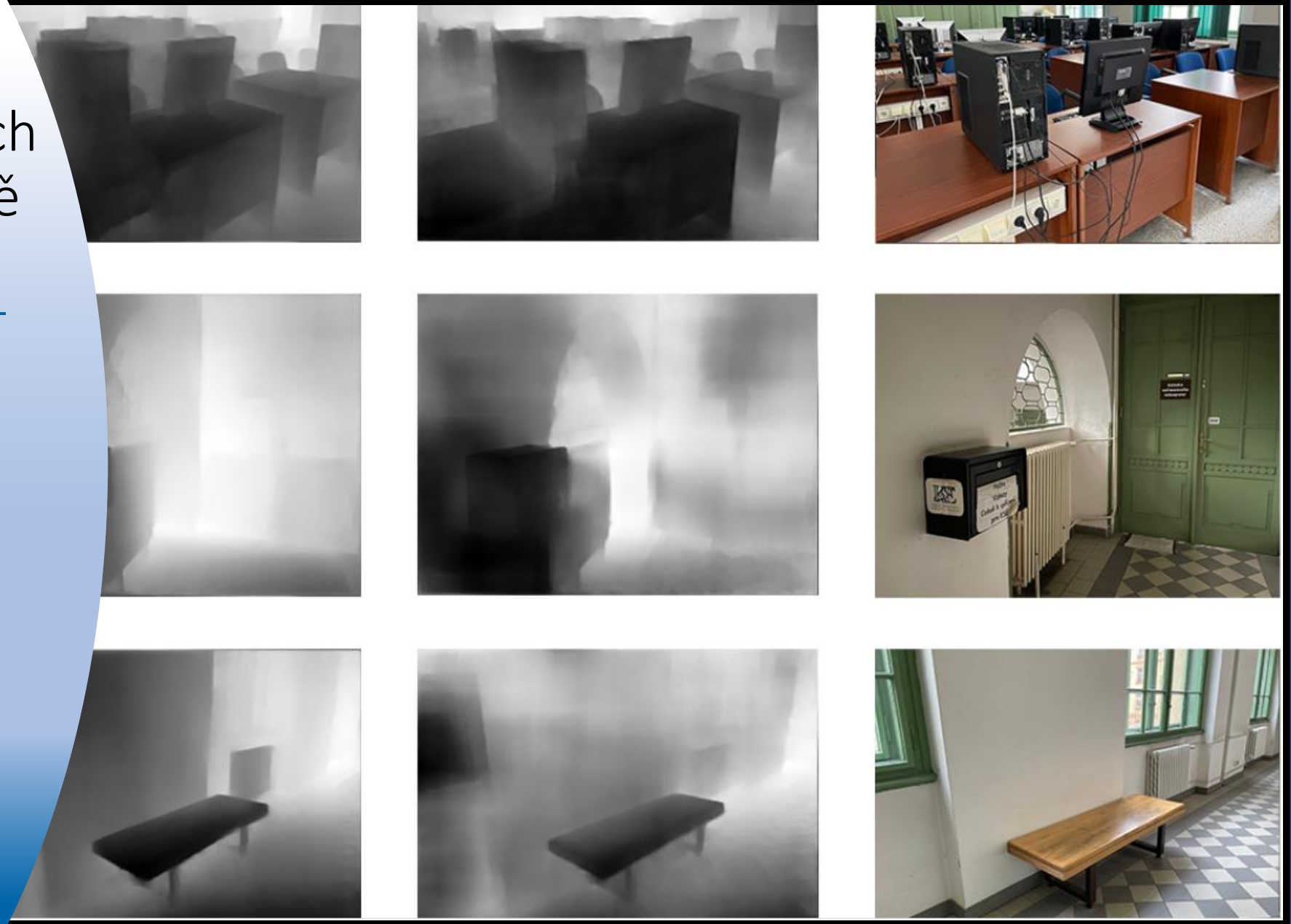
- Implementace CNN v prostředí Matlab
- Srovnání různých webscrapingových metod pro získání obrazových dat z webu
- Srovnání různých modelů CNN

Aplikace neuronových sítí při vývoji her

- Využití herního engine Unity
- Umělá inteligence pro hru Mario Bros
- Jednoduchý fotbal

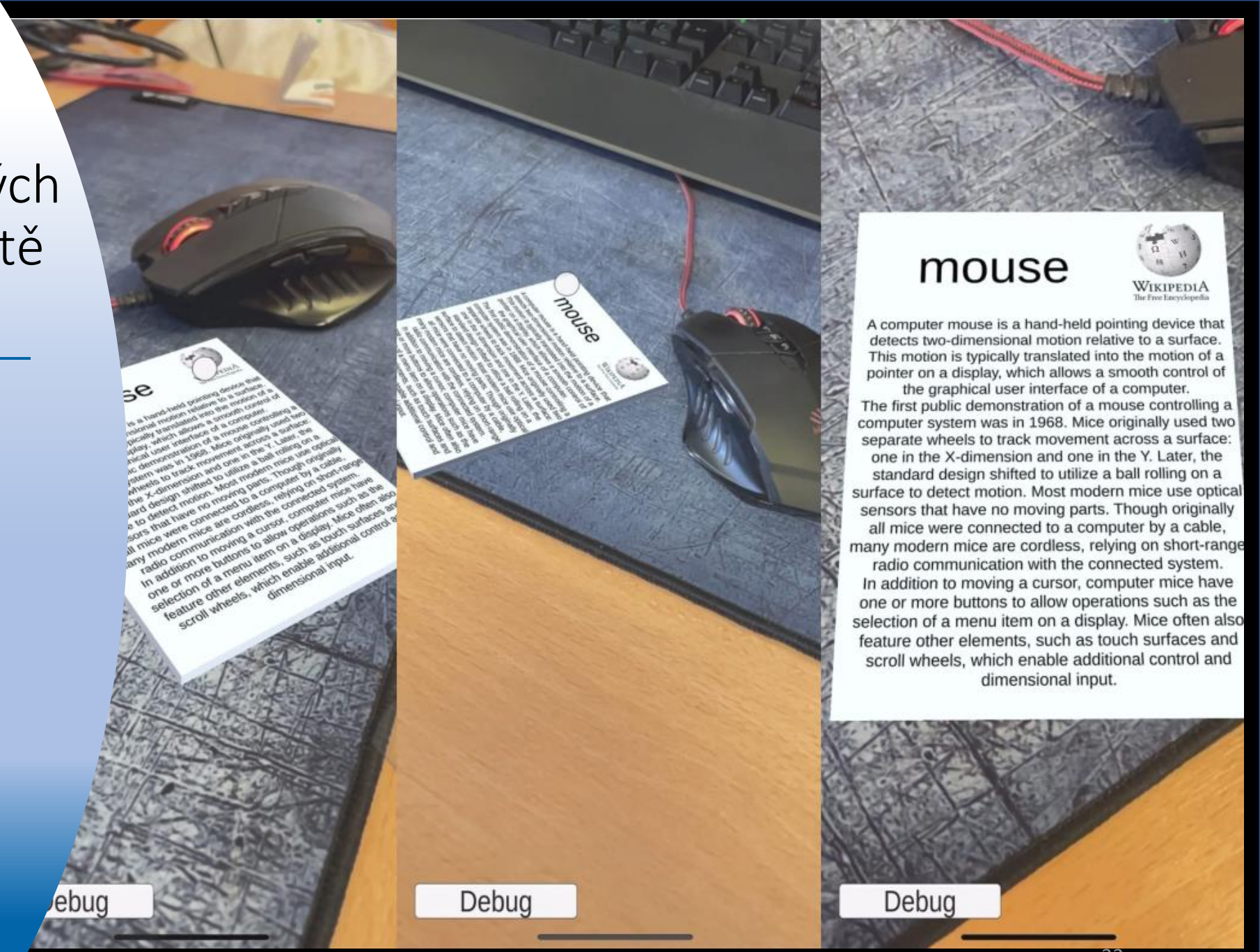
Aplikace neuronových sítí v rozšířené realitě

- Odhad hloubky scény na základě jedné fotografie
- Značná redukce počtu parametrů a tím doby trénování
- Výsledky srovnatelné s modely jiných autorů



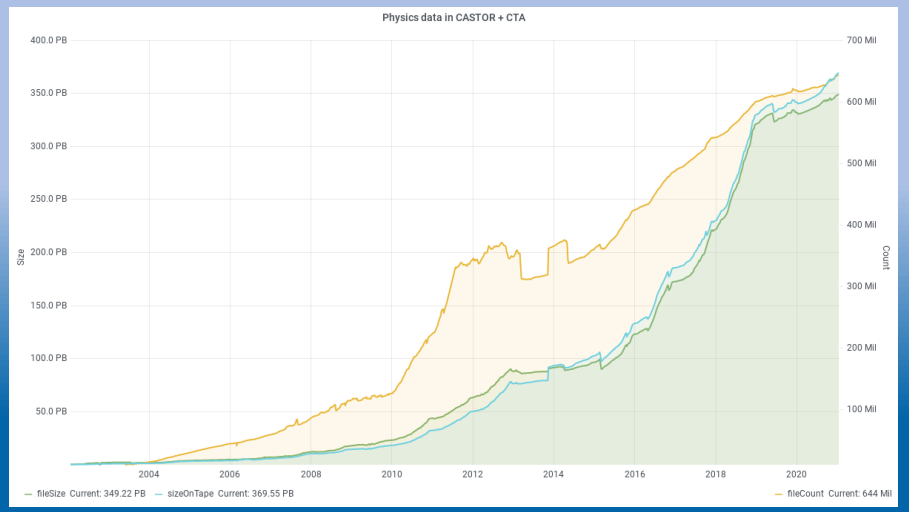
Aplikace neuronových sítí v rozšířené realitě

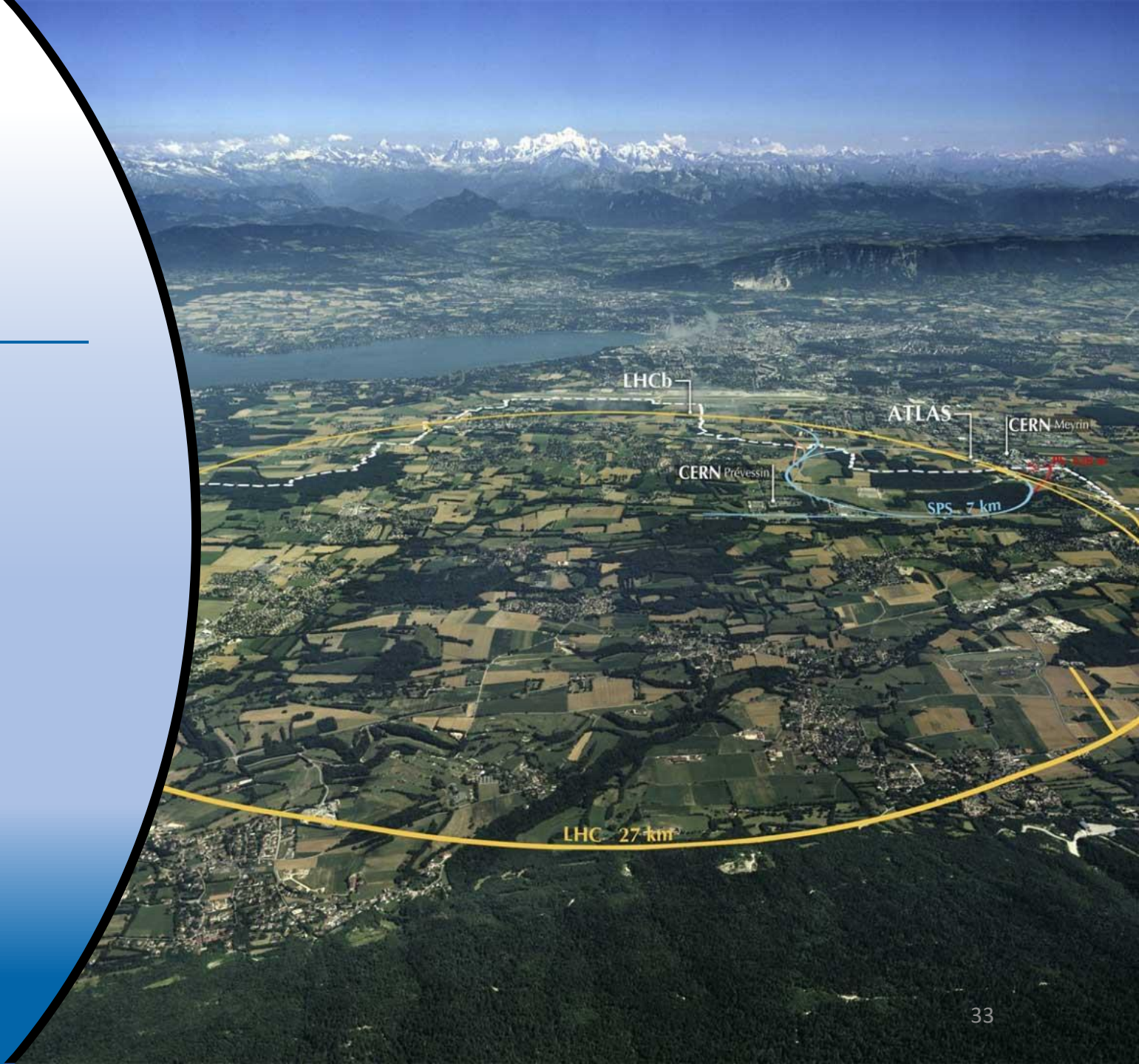
- Využití konvoluční neuronové sítě pro rozpoznání objektů
- Mobilní aplikace pro iOS
- Využití engine Unity a knihovny Tensor Flow Lite, AR foundation



CERN

- Mezinárodní výzkumné centrum v Ženevě
- Možnosti stáží a práce na místních experimentech
- Domov velkého množství experimentů s potřebou IT odborníků





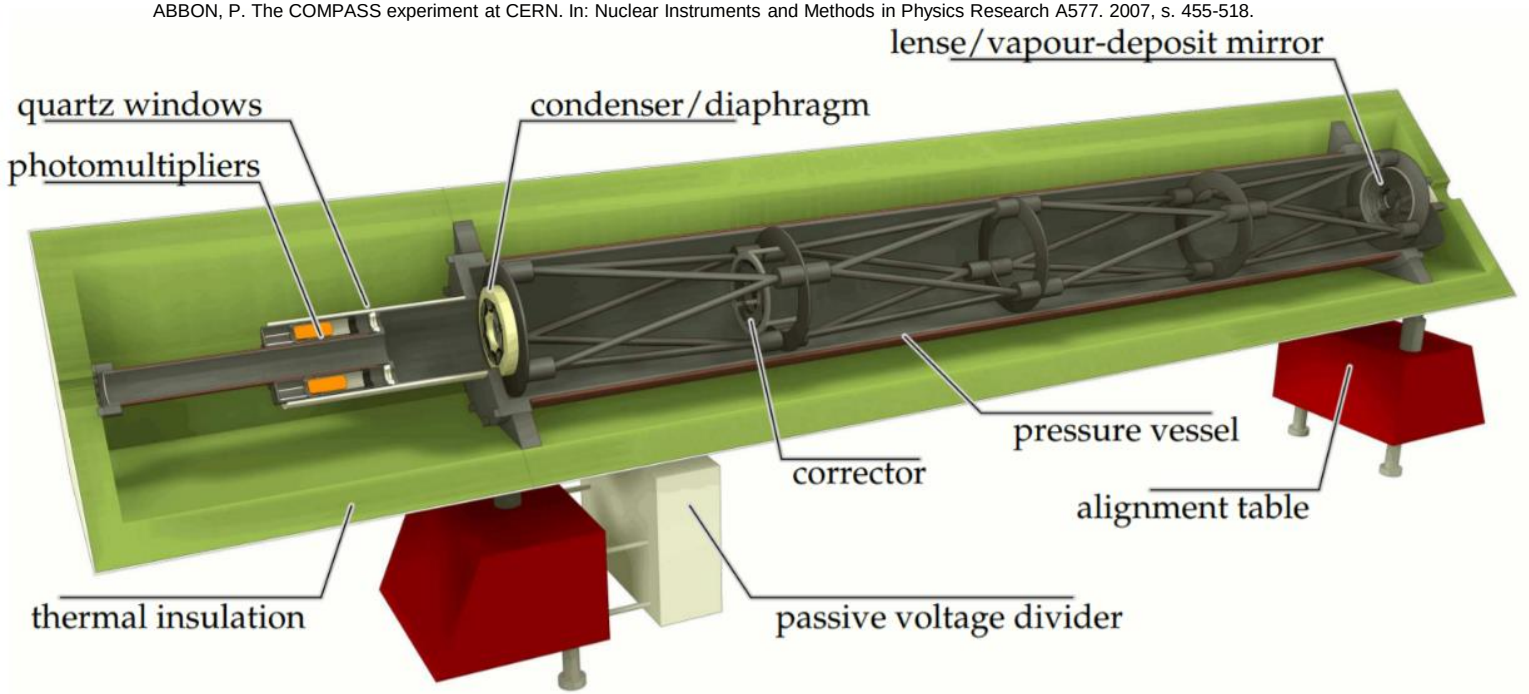
CERN – Experimenty AMBER a COMPASS

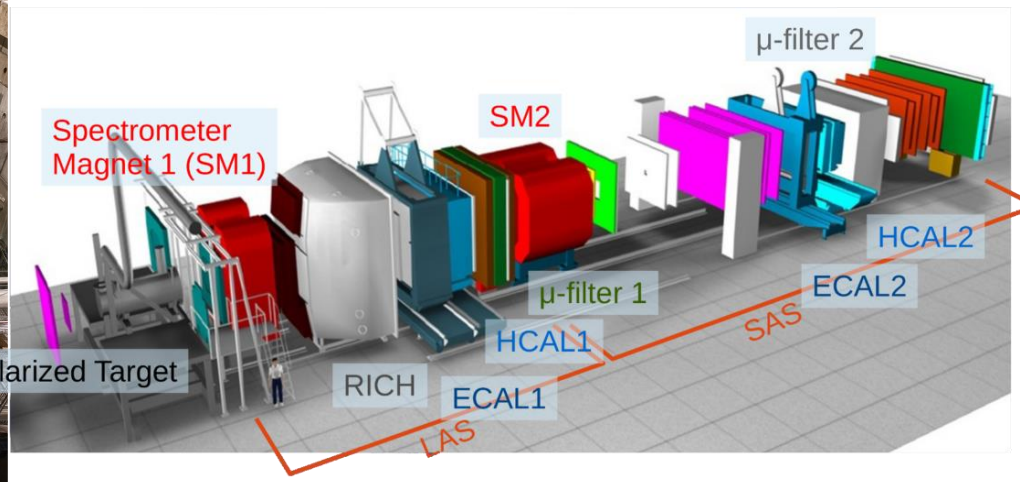
- Experimenty s pevným terčem na urychlovači SPS v CERN (největší nadzemní – 60 m)

  Úlohy:

  - Rozpoznávání vzorů
  - Detekce anomálií
  - Klasifikace
  - Rychlé triggery
  - Rekonstrukce dat
- Mnoho dalších experimentů i mimo urychlovače (OSQAR, NA64)

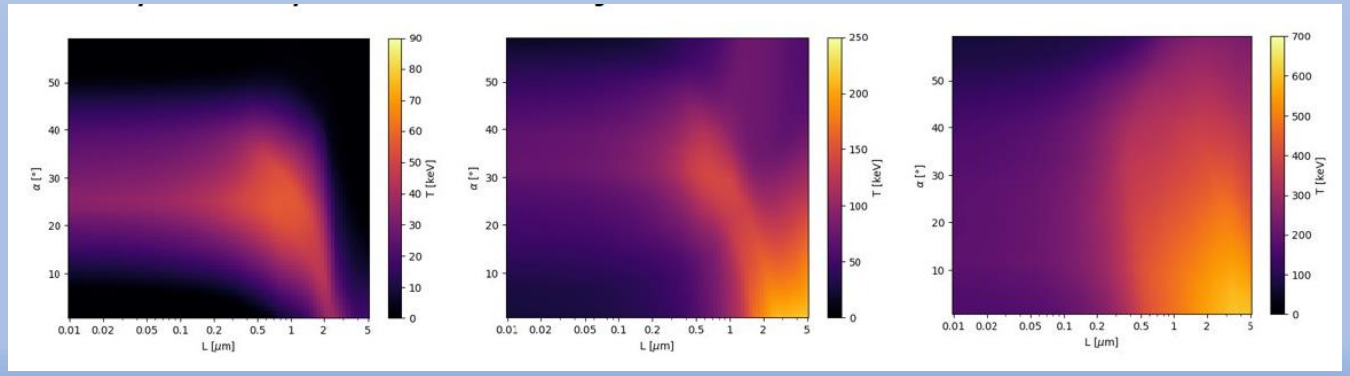
ABBON, P. The COMPASS experiment at CERN. In: Nuclear Instruments and Methods in Physics Research A577. 2007, s. 455-518.





Další tvorba studentů

- Predikce vývoje časových řad (akcie, kurzy měn, počasí,...)
- Modelování ekonomických systémů
- Fyzikální aplikace
  - Např. modelování interakce laserového záření s plazmatem s využitím neuronových sítí:



Kde se dozvím více o hlubokém učení?

- Interaktivní ukázka MLP: http://playground.tensorflow.org/
- Ukázkové Jupyter Notebooky, datasety: https://www.kaggle.com
- Dokumentace Kerasu: https://keras.io/api/

Nyní nás čeká ještě krátká přednáška o studiu na FJFI ČVUT a o nejbližších akcích, které pořádáme pro středoškolské studenty