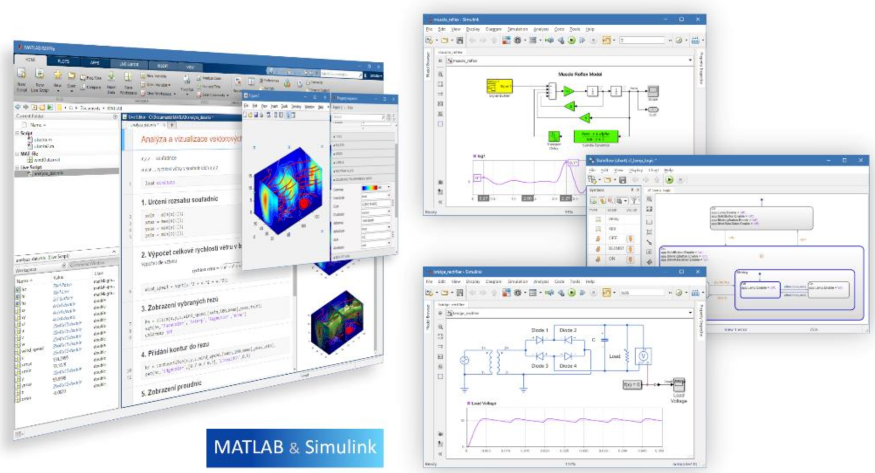
HUMUSOFT®

10.9.2020 Brno

TCC 2020

Deep learning

nové možnosti pro začátečníky i pokročilé uživatele

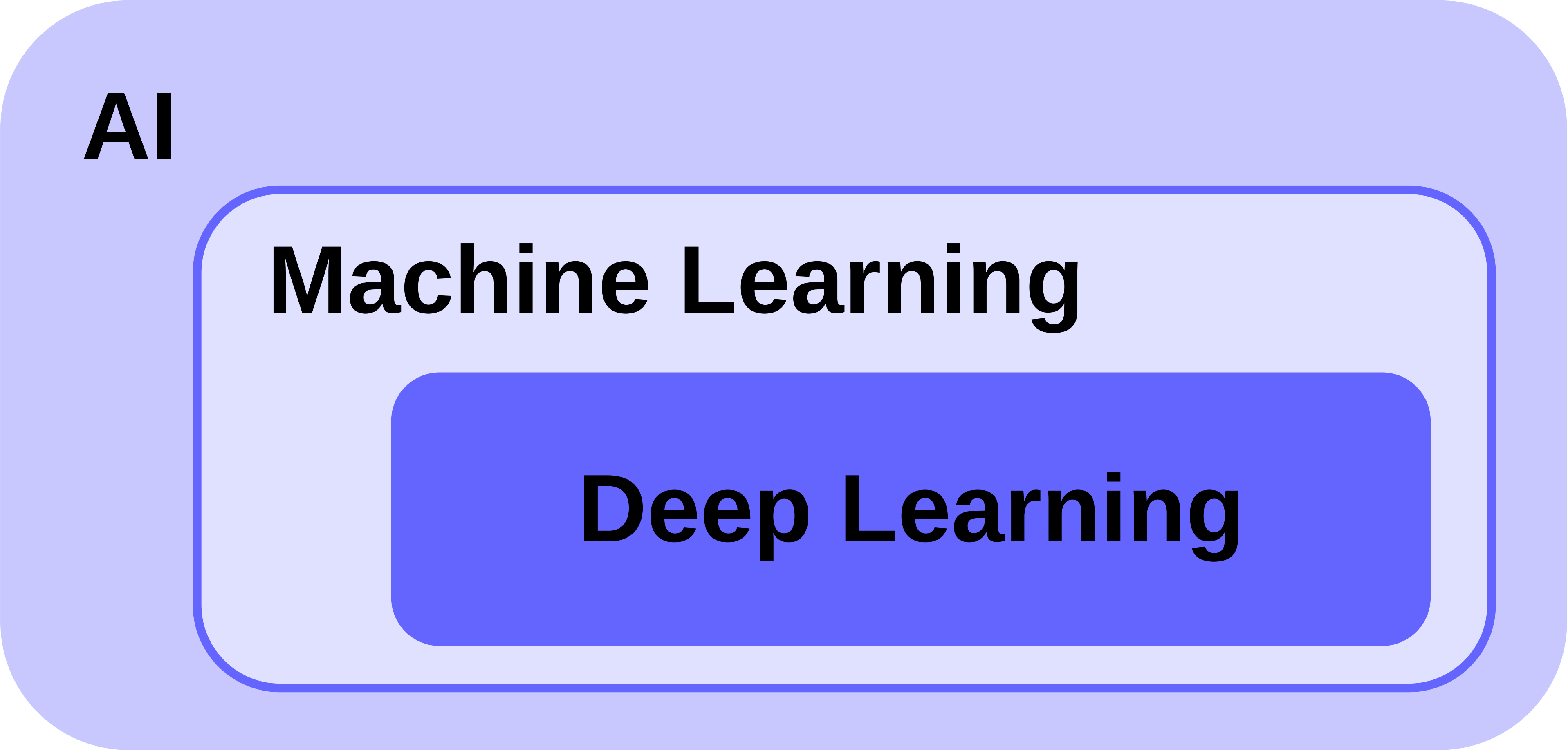


Jaroslav Jirkovský
jirkovsky@humusoft.cz

www.humusoft.cz
info@humusoft.cz

www.mathworks.com

AI, Machine Learning, and Deep Learning



AI is Just One Part of System Development Workflow

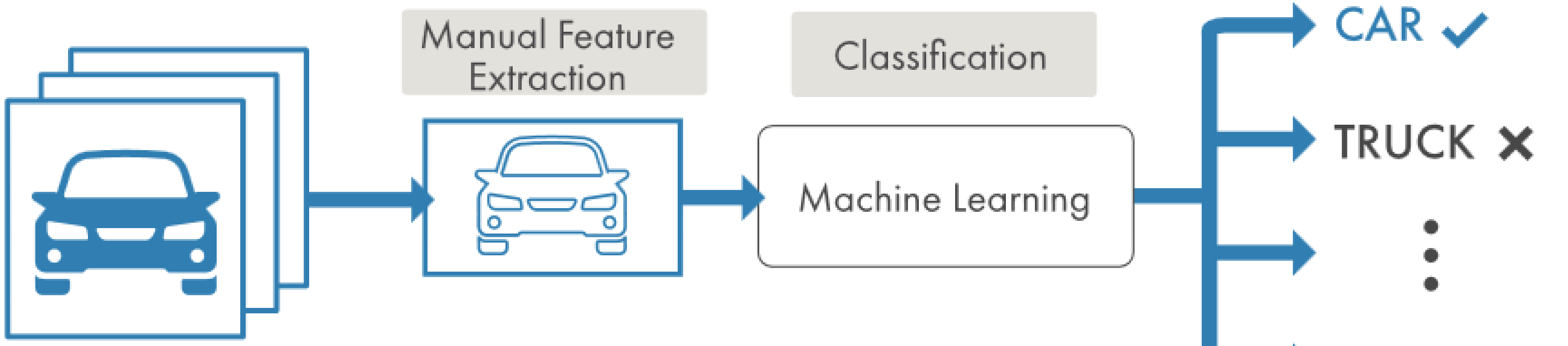
with MATLAB and Simulink

Access Data	Analyze Data	Develop	Deploy
Sensors	Data exploration	AI model	Desktop apps
Files	Preprocessing	Algorithm development	Enterprise systems
Databases	Domain-specific algorithms	Modeling & simulation	Embedded devices

What is Machine Learning ?

Machine learning uses data and produces a program to perform a task

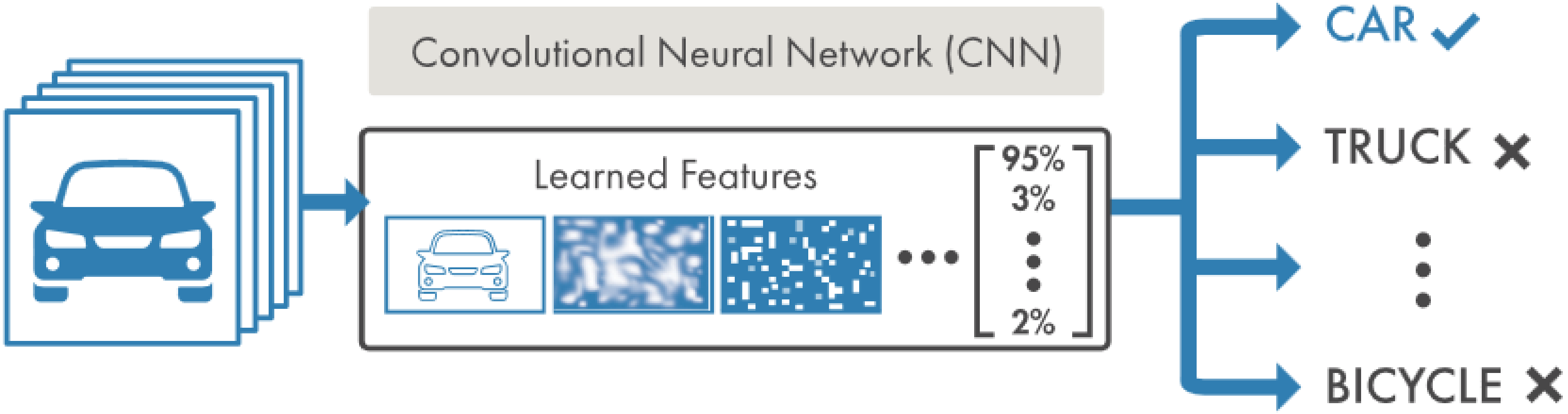
MACHINE LEARNING



What is Deep Learning ?

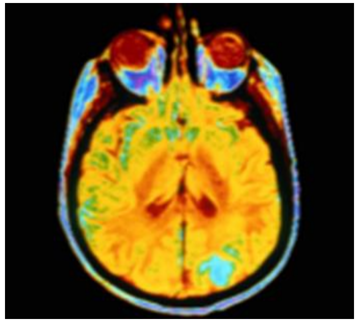
Deep learning performs end-end learning by learning features, representations and tasks directly from images, text and sound

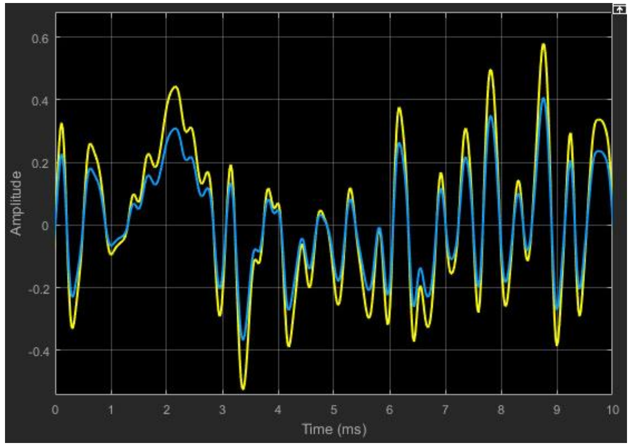
DEEP LEARNING



Deep Learning is Ubiquitous

- Computer Vision
- Signal Processing
- Robotics & Controls
- …



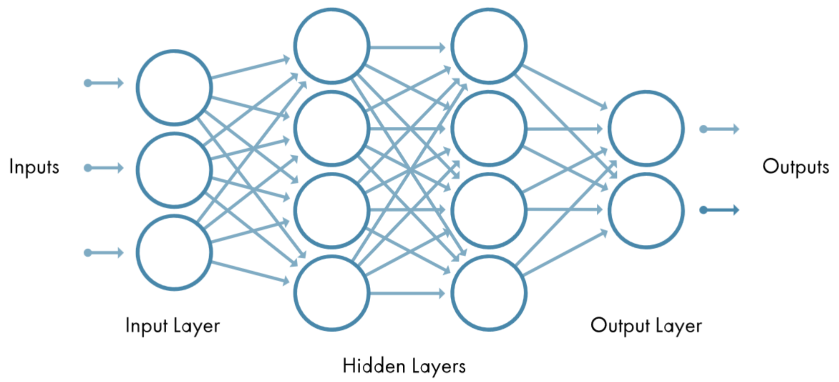


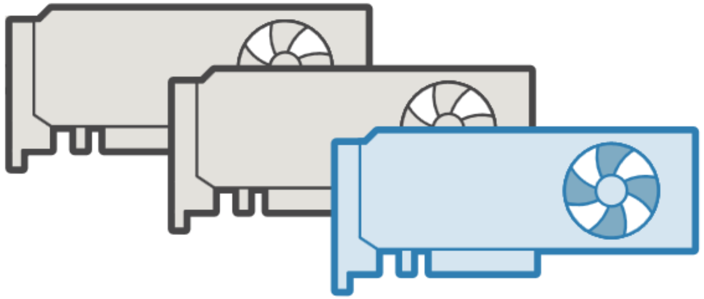
MATLAB for Deep Learning

- Network Architectures and Algorithms
- Training and Visualization
- Access the Latest Pretrained Models
- Scaling and Acceleration
- Handling Large Sets of Images

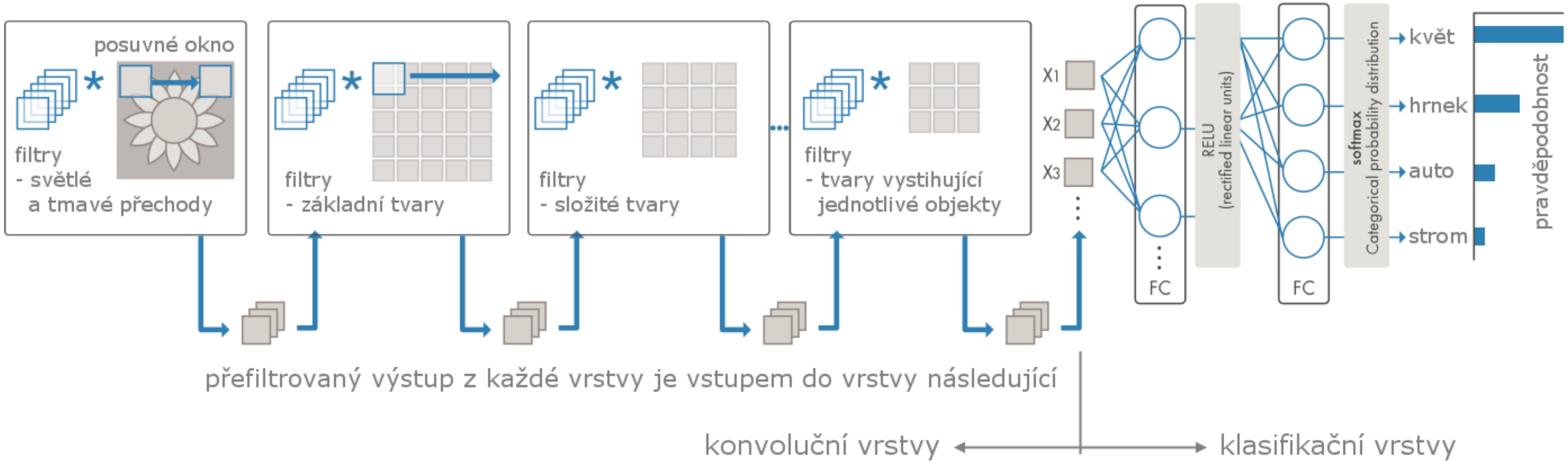
- Classification and Regression
- Object Detection
- Semantic Segmentation

- Embedded Deployment

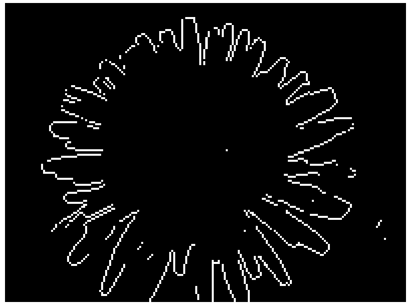




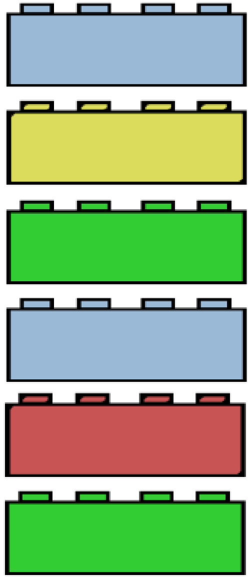
Convolutional Neural Networks (CNN)

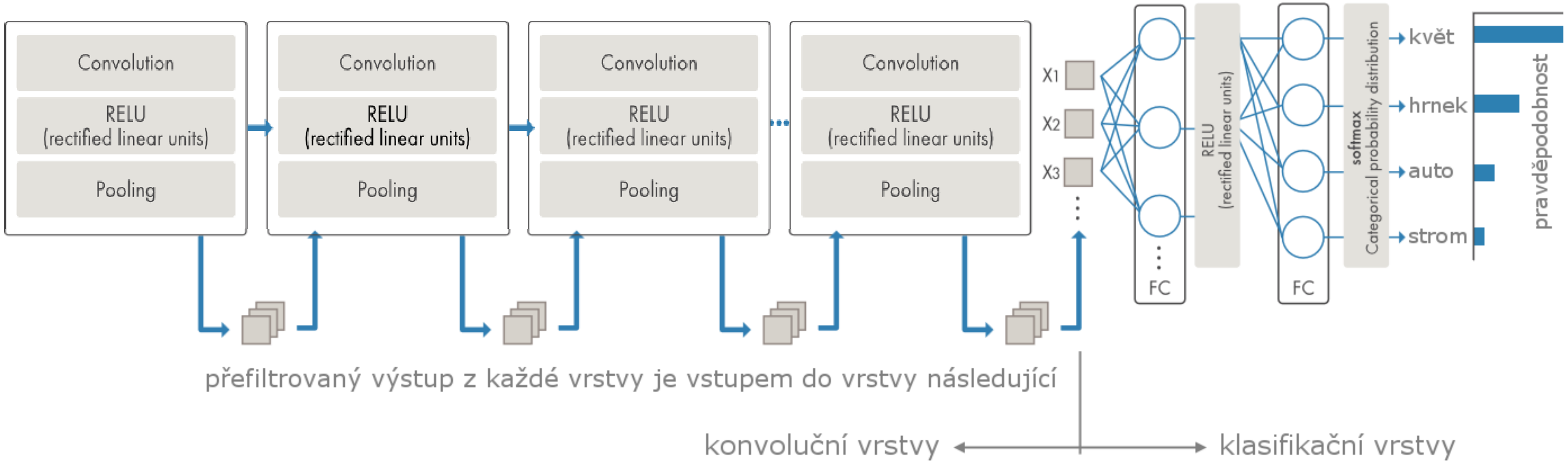


What do filters do?



Convolutional Neural Networks (CNN)





Convolution

RELU (rectified linear units)

Pooling

Convolution

RELU (rectified linear units)

Pooling

Convolution

RELU (rectified linear units)

Pooling

Convolution

RELU (rectified linear units)

Pooling

X1

X2

X3

FC

RELU (rectified linear units)

FC

softmax Categorical probability distribution

květ

hrnek

auto

strom

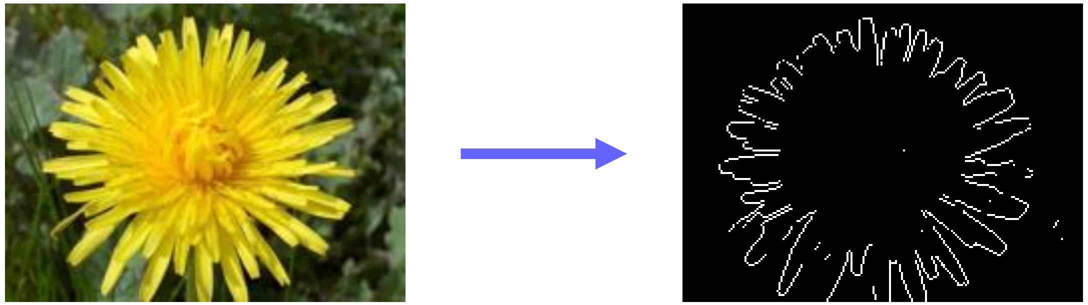
pravděpodobnost

přefiltrovaný výstup z každé vrstvy je vstupem do vrstvy následující

konvoluční vrstvy

klasifikační vrstvy

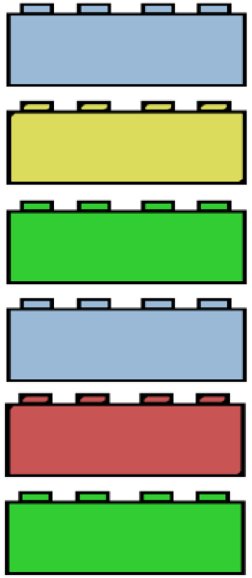
What do filters do?



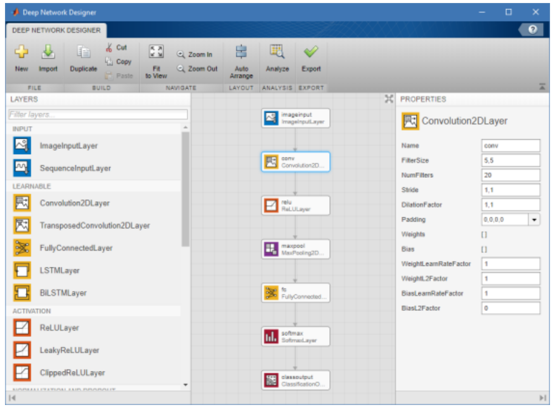
Great for classification:

- Convolution Layer
- ReLU Layer
- Max Pooling Layer

Deep Neural Networks in MATLAB – 3 Approaches



Deep Network Designer



Convolution2DLayer

Name	conv
FilterSize	5,5
NumFilters	20
Stride	1,1
DilationFactor	1,1
Padding	0,0,0,0
Weights	[]
Bias	[]
WeightLearnRateFactor	1
WeightL2Factor	1
BiasLearnRateFactor	1
BiasL2Factor	0

Standard Framework

```
layers = [imageInputLayer([28 28 1])
          convolution2dLayer(5,20)
          reluLayer()
          maxPooling2dLayer(2,'Stride',2)
          fullyConnectedLayer(10)
          softmaxLayer()
          classificationLayer()];
```

Extended Framework

training using APP

```
options = trainingOptions('sgdm');
convnet = trainNetwork(data,layers,opts);
```

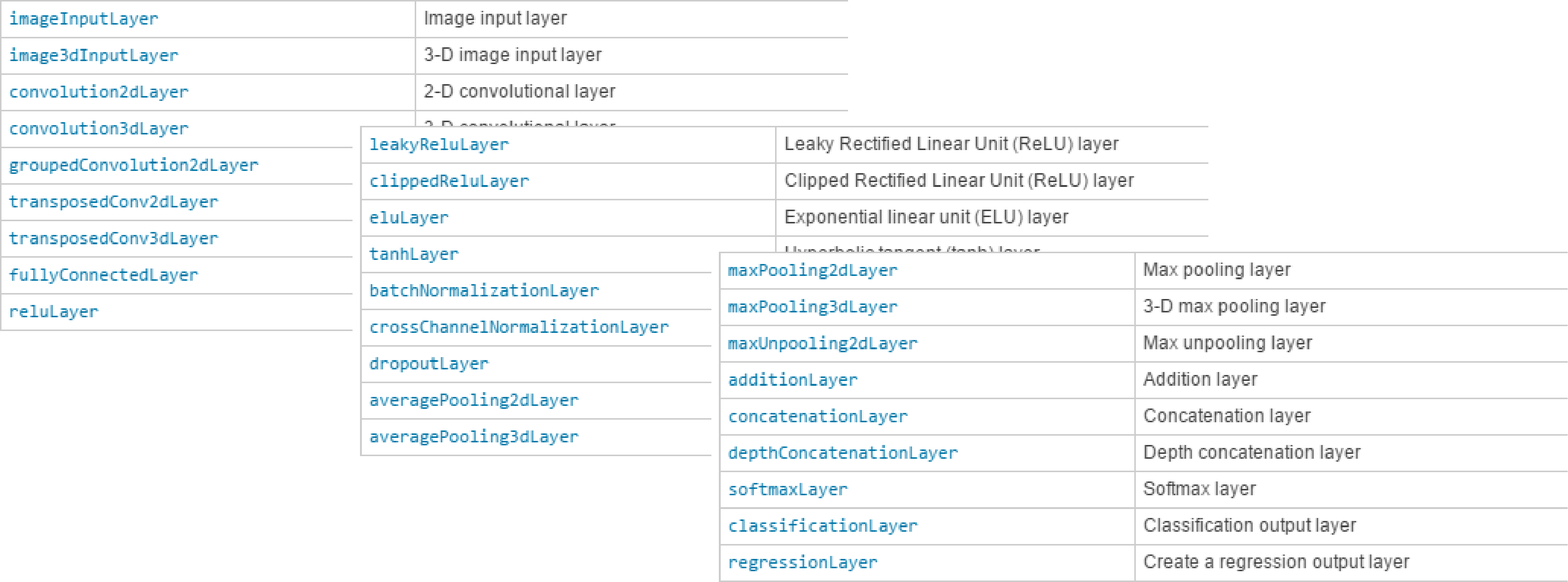
```
results = classify(convnet,newData);
```

- for most deep learning tasks

- custom training loops
- automatic differentiation
- shared weights
- custom loss functions
- ...

- GANs, Siamese networks, ...

>30 Layers



imageInputLayer	Image input layer
image3dInputLayer	3-D image input layer
convolution2dLayer	2-D convolutional layer
convolution3dLayer	3-D convolutional layer
groupedConvolution2dLayer	
transposedConv2dLayer	
transposedConv3dLayer	
fullyConnectedLayer	
reluLayer	

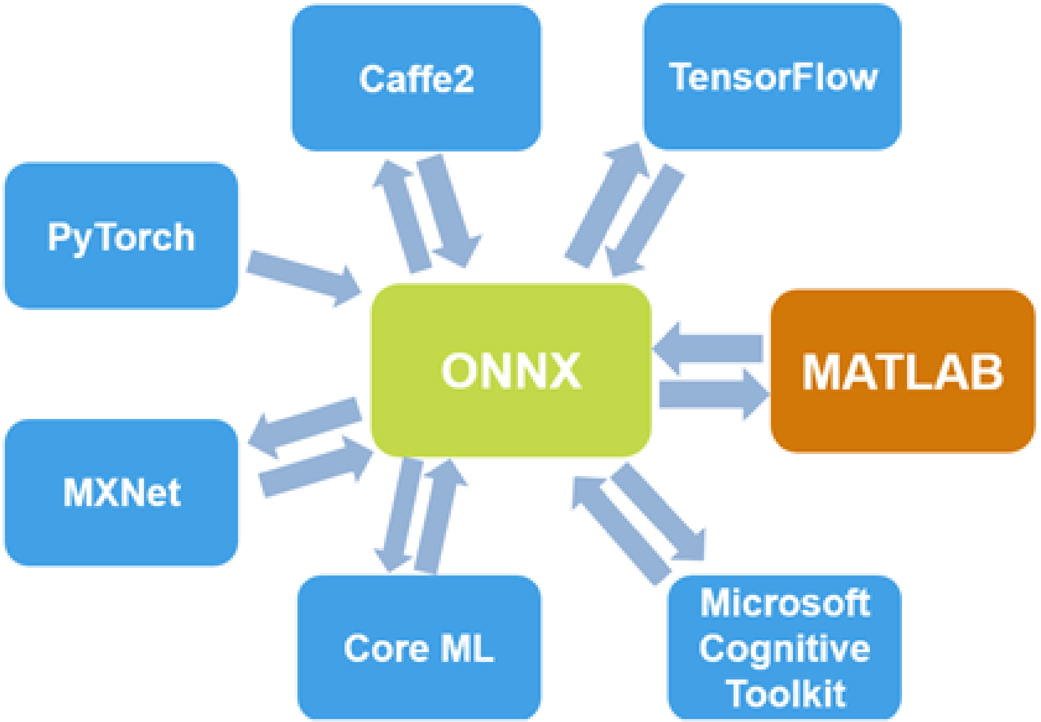
leakyReluLayer	Leaky Rectified Linear Unit (ReLU) layer
clippedReluLayer	Clipped Rectified Linear Unit (ReLU) layer
eluLayer	Exponential linear unit (ELU) layer
tanhLayer	Hyperbolic tangent (tanh) layer
batchNormalizationLayer	
crossChannelNormalizationLayer	
dropoutLayer	
averagePooling2dLayer	
averagePooling3dLayer	

maxPooling2dLayer	Max pooling layer
maxPooling3dLayer	3-D max pooling layer
maxUnpooling2dLayer	Max unpooling layer
additionLayer	Addition layer
concatenationLayer	Concatenation layer
depthConcatenationLayer	Depth concatenation layer
softmaxLayer	Softmax layer
classificationLayer	Classification output layer
regressionLayer	Create a regression output layer

- **Author custom layers in MATLAB using the Custom Layer API**
  - **including automatic differentiation**

Transfer Learning using Pre-Trained Networks

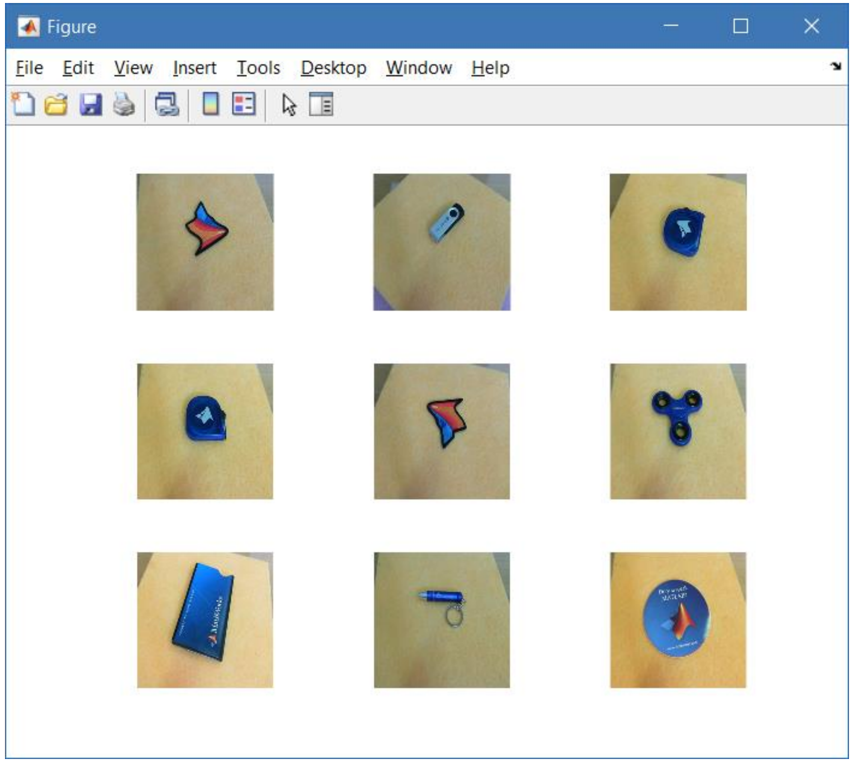
- **Pre-Trained Networks**
  - AlexNet
  - VGG-16 and VGG-19
  - GoogLeNet
  - ResNet-50 and ResNet-101
  - Inception-v3
  - Inception-ResNet-v2
  - SqueezeNet
  - and more …



- **ONNX Model Converter**

Example: Fine-tune a pre-trained model (transfer learning)

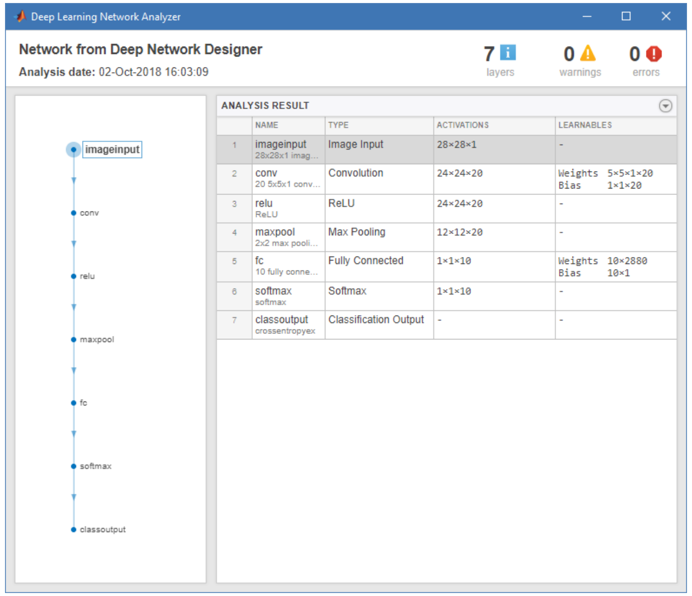
- https://www.mathworks.com/help/deeplearning/gs/get-started-with-deep-network-designer.html



Training, Validation and Visualization

- **Network Analyzer (`analyzeNetwork`)**
  - – find problems in network architectures before training

- **Monitor training progress**
  - – plots for accuracy, loss, validation metrics, and more
- **Automatically validate network performance**
  - – stop training when the validation metrics stop improving
- **Perform hyperparameter tuning**
  - – using Bayesian optimization



Deep Learning Network Analyzer

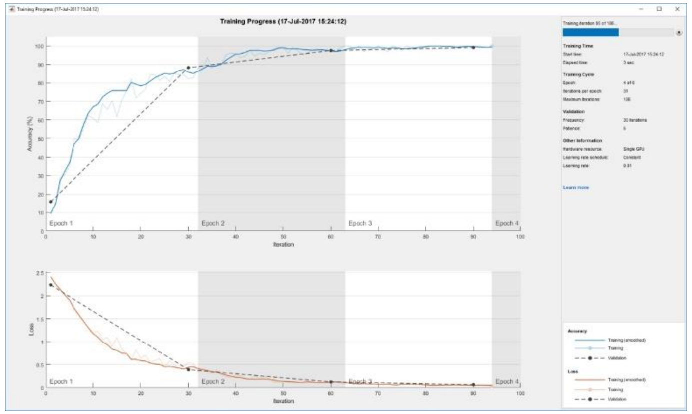
Network from Deep Network Designer

Analysis date: 02-Oct-2018 16:03:09

7 layers | 0 warnings | 0 errors

ANALYSIS RESULT

	NAME	TYPE	ACTIVATIONS	LEARNABLES
1	imageinput 28x28x1 imag...	Image Input	28×28×1	-
2	conv 20 5x5x1 conv...	Convolution	24×24×20	Weights 5×5×1×20 Bias 1×1×20
3	relu ReLU	ReLU	24×24×20	-
4	maxpool 2x2 max pooli...	Max Pooling	12×12×20	-
5	fc 10 fully conne...	Fully Connected	1×1×10	Weights 10×2880 Bias 10×1
6	softmax softmax	Softmax	1×1×10	-
7	classoutput crossentropyex	Classification Output	-	-



Training Progress (17-Jul-2017 15:24:12)

Training iteration 95 of 186...

Training Time
Start time: 17-Jul-2017 15:24:12
Elapsed time: 3 sec

Training Cycle
Epoch: 4 of 6
Iterations per epoch: 31
Maximum iterations: 186

Validation
Frequency: 30 iterations
Patience: 5

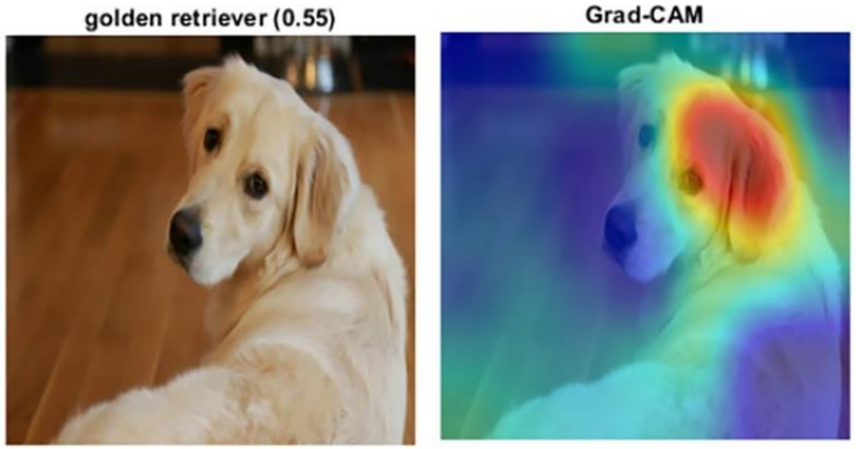
Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 0.01

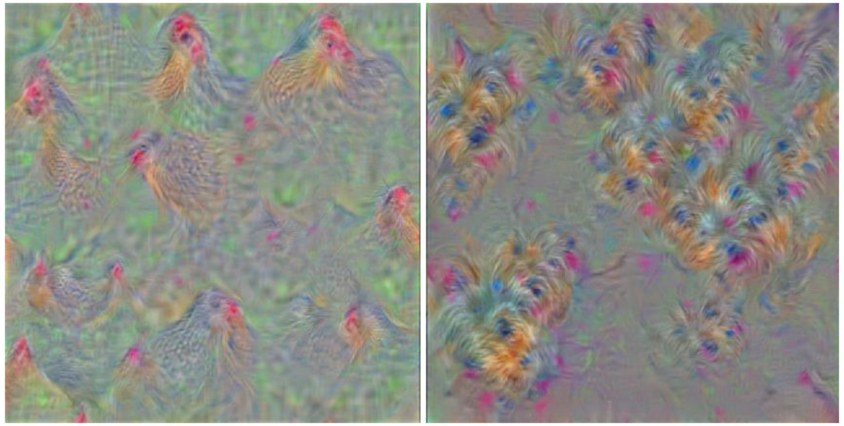
Debugging and Visualization

- Visualize activations and filters from intermediate layers

- CAM (Class Activation Mapping)
- Grad-CAM
- Occlusion sensitivity maps

- Deep Dream visualization





Handling Large Sets of Images

- Use `imageDataStore`
  - easily read and process large sets of images
- Access data stored in
  - local files
  - networked storage
  - databases
  - big data file systems
- Efficiently resize and augment image data
  - increase the size of training datasets
  - `imageDataAugmenter`, `augmentedImageDatastore`

Customizations (Extended Framework)

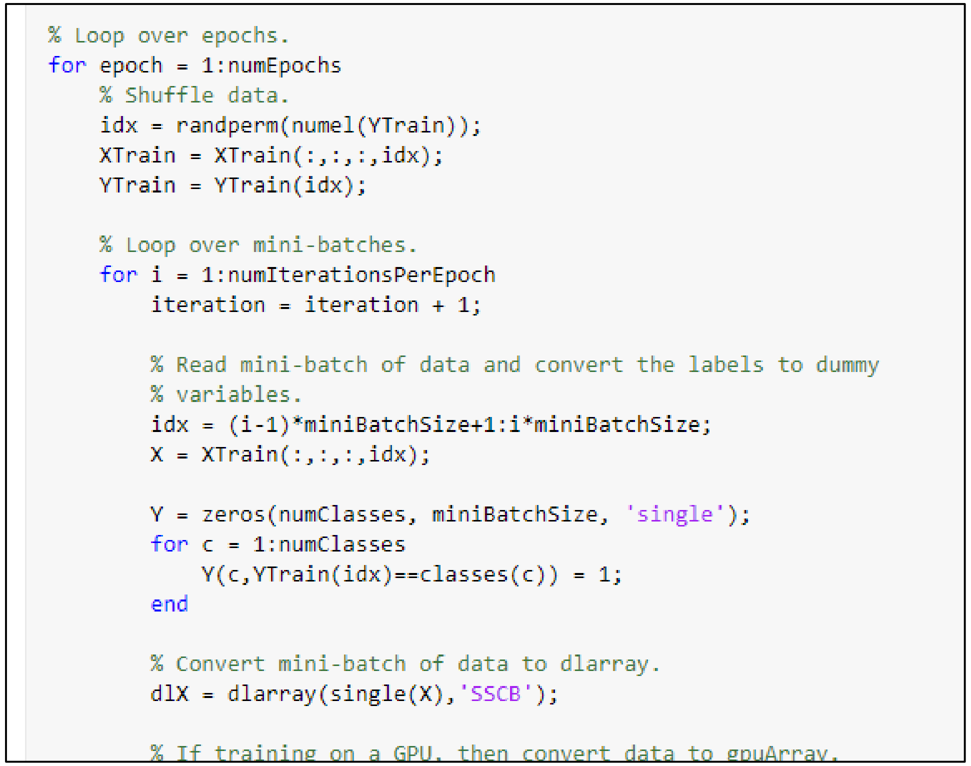
- **Define and train complex networks using**
  - custom training loops
  - automatic differentiation
  - shared weights
  - custom loss functions
- **Custom layers support**
  - define new layers, including layers with multiple inputs and outputs
- **Multi-Input, Multi-Output Networks**
  - create and train networks with multiple inputs and multiple outputs
- **Build advanced network architectures**
  - GANs, Siamese networks, attention networks, ...

Using Custom Training Loops

1. Define your network
   - *lgraph* object, same as standard approach, without classification layer
2. Convert network object to *dlnetwork* object
3. Define your custom training options
   - as a set of variables, not options-object
4. Define you custom training loop
   - for loops over number of epochs and iterations
   - read data, convert to *dlarray* (and *gpuArray* for GPU computing)
   - calculate model *gradients* and *loss* (use automatic differentiation)
   - run *solver* for network update
5. Use the trained network
   - *predict* function, convert to class selection

Example: Extended Framework

- https://www.mathworks.com/help/deeplearning/ug/train-network-using-custom-training-loop.html



```
% Loop over epochs.
for epoch = 1:numEpochs
    % Shuffle data.
    idx = randperm(numel(YTrain));
    XTrain = XTrain(:,:,:,idx);
    YTrain = YTrain(idx);

    % Loop over mini-batches.
    for i = 1:numIterationsPerEpoch
        iteration = iteration + 1;

        % Read mini-batch of data and convert the labels to dummy
        % variables.
        idx = (i-1)*miniBatchSize+1:i*miniBatchSize;
        X = XTrain(:,:,:,idx);

        Y = zeros(numClasses, miniBatchSize, 'single');
        for c = 1:numClasses
            Y(c,YTrain(idx)==classes(c)) = 1;
        end

        % Convert mini-batch of data to dlarray.
        dlX = dlarray(single(X),'SSCB');

        % If training on a GPU, then convert data to gpuArray.
```

Generative Adversarial Network (GAN)

- Generate data with similar characteristics as the input real data
- Two networks that train together

- Generator
  - input vector of random values
  - generates data with the same structure as the training data

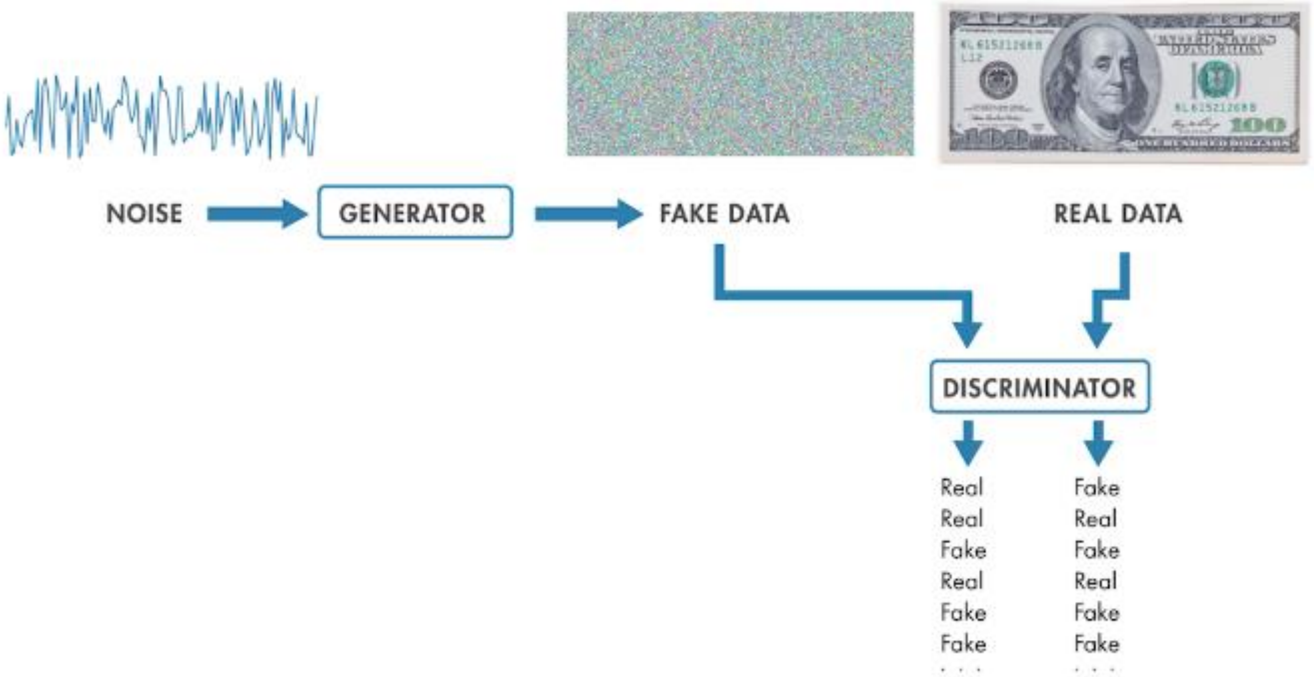
- Discriminator
  - observations from the training data, and generated data
  - classify the observations as "real" or "generated".

Generative Adversarial Network (GAN)

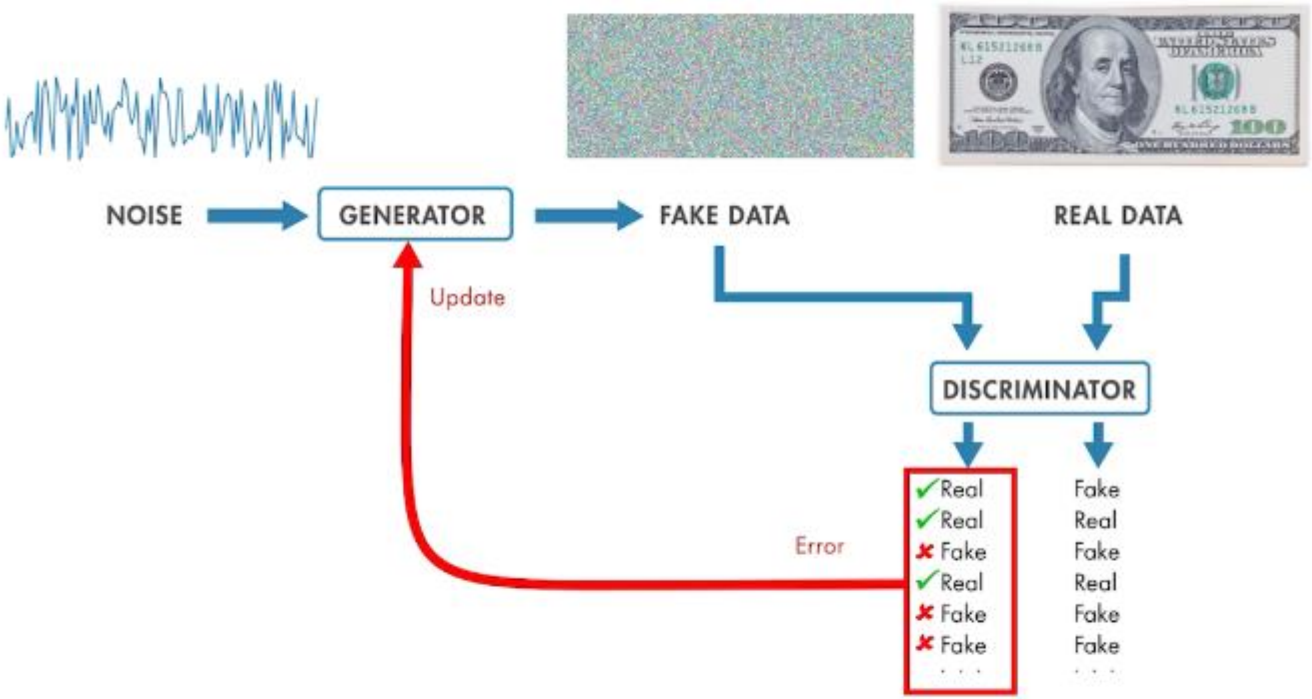
GENERATOR

DISCRIMINATOR

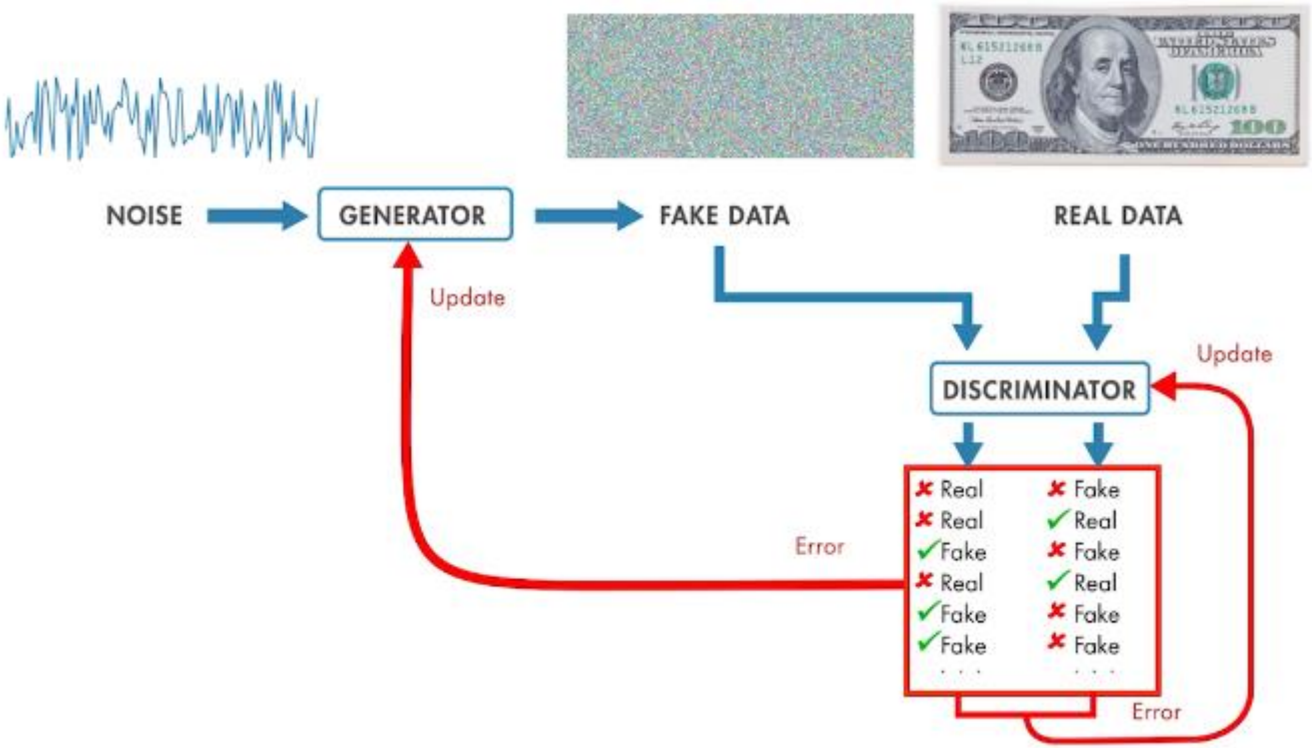
Generative Adversarial Network (GAN)



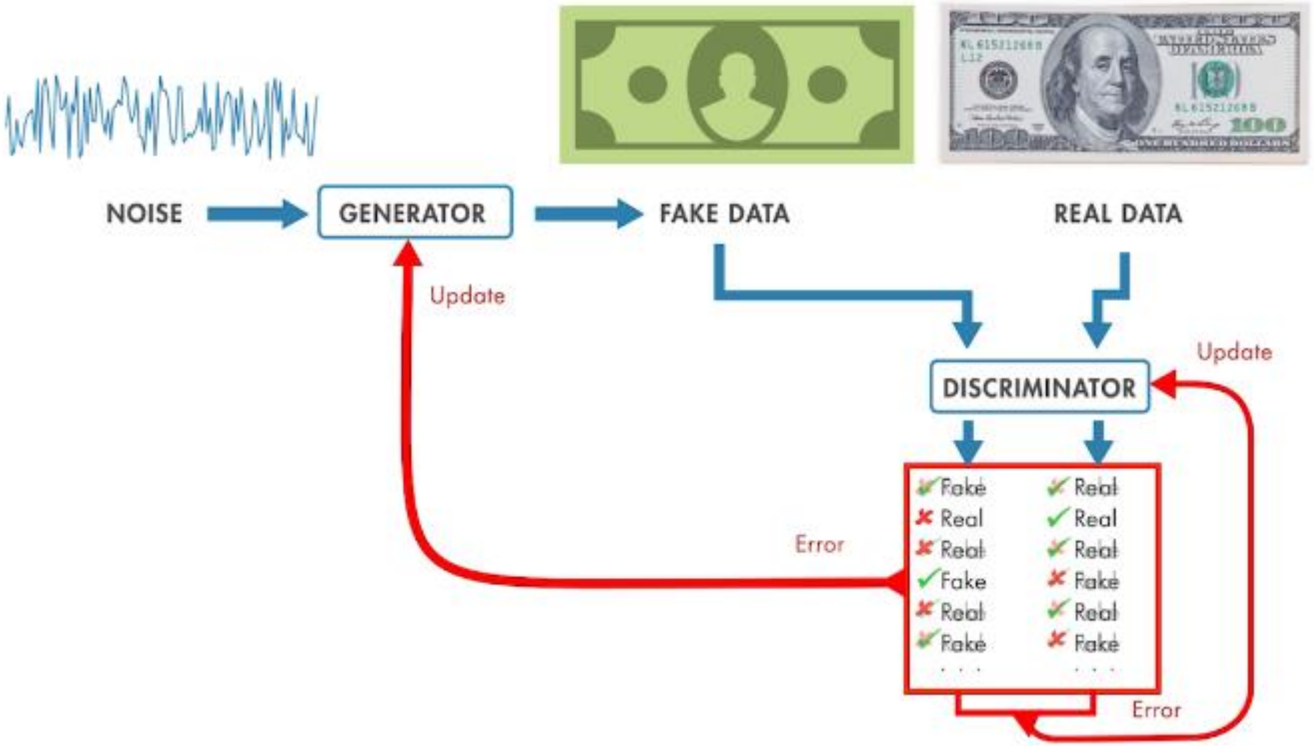
Generative Adversarial Network (GAN)



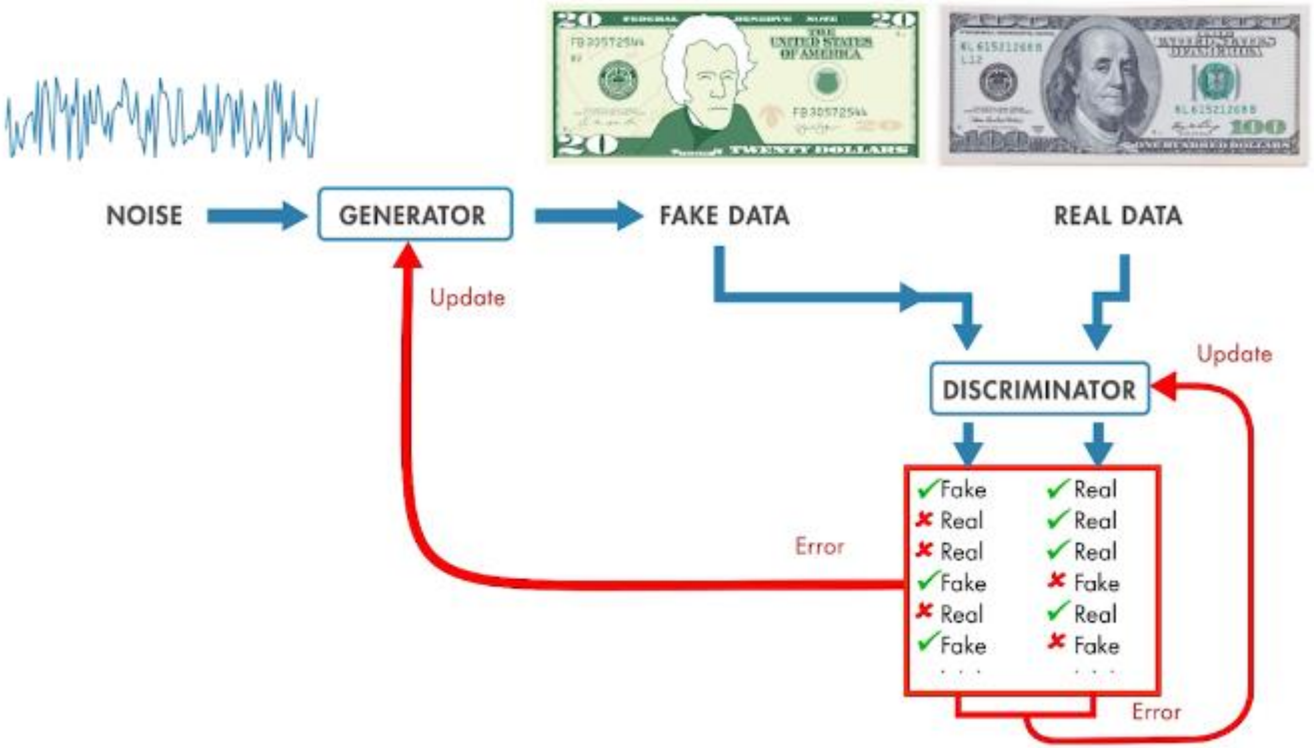
Generative Adversarial Network (GAN)



Generative Adversarial Network (GAN)

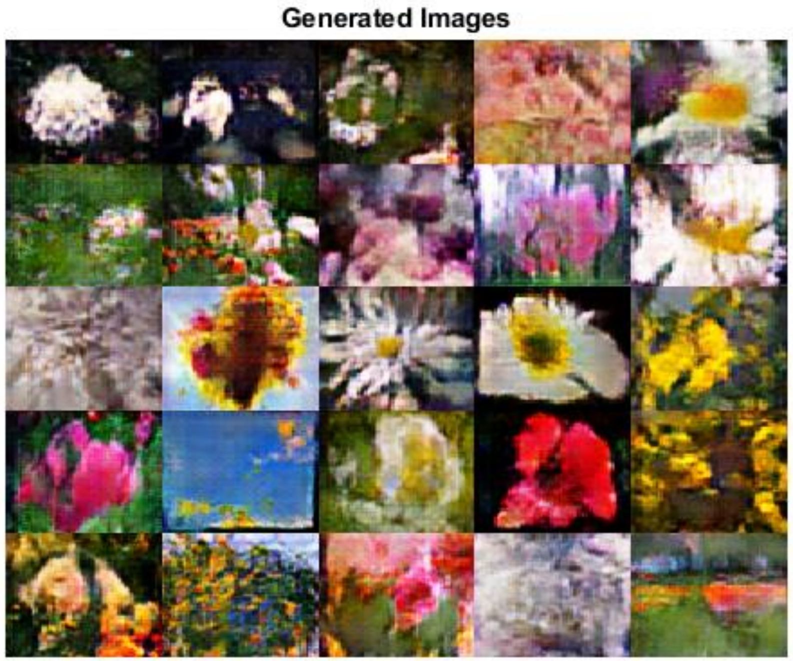


Generative Adversarial Network (GAN)



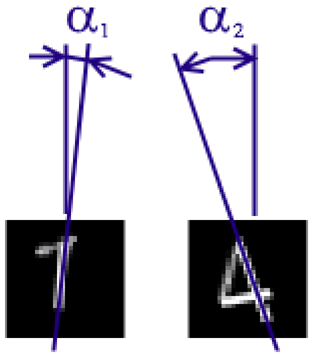
Example: Generative Adversarial Network

- https://www.mathworks.com/help/deeplearning/ug/train-generative-adversarial-network.html

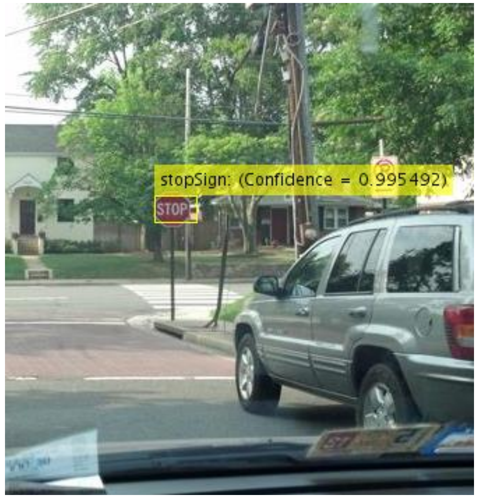


Other deep learning tasks with images

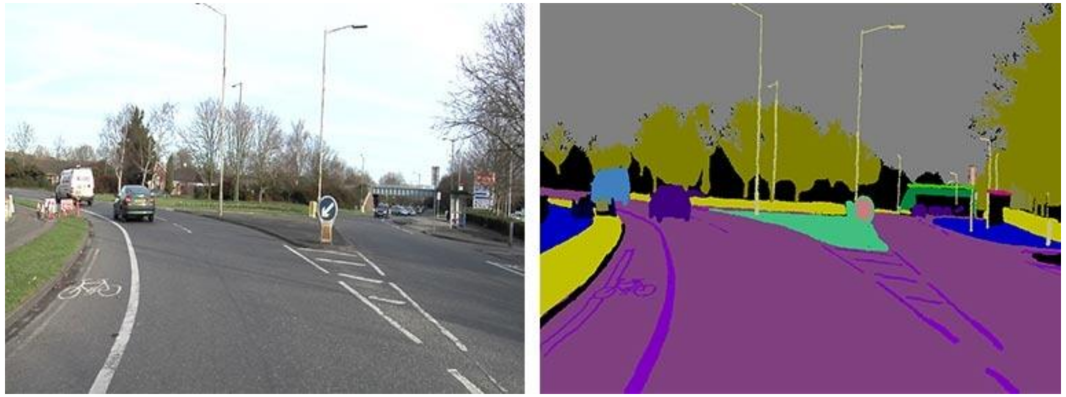
- **Regression**
  - predict continuous variable from the image



- **Object Detection**
  - recognizing and locating the object in a scene
  - multiple objects in one image

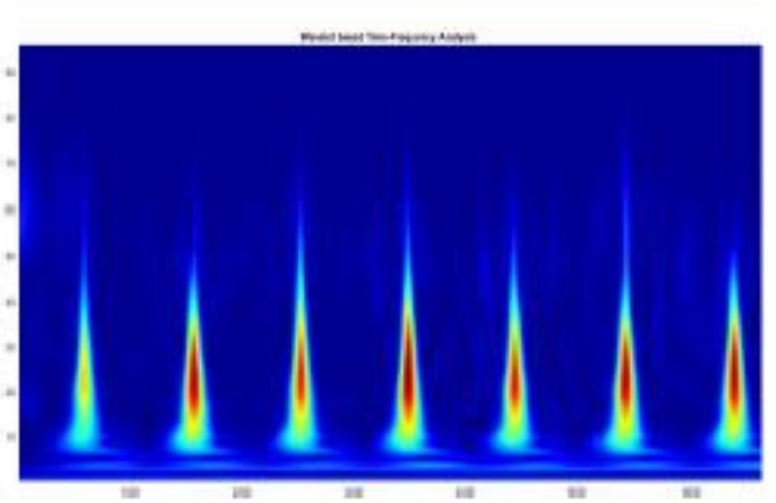


- **Semantic Segmentation**
  - classify individual pixels



Deep learning for signal processing

- **Leverage CNNs with signals**
  - **„convert“ signal into image using time-frequency representations**
    - how spectral content of signal evolves over time
  - **many time-frequency representations available**
    - spectrogram, cwt, stft
    - doc "Time-Frequency Gallery"



- **Special network layers for signals – LSTM networks**
  - **classification and prediction**

Multi-Platform Deployment

- Deploy deep learning models anywhere
  - CUDA
  - C code
  - enterprise systems
  - or the cloud
- Generate code that leverages optimized libraries
  - Intel® (MKL-DNN)
  - NVIDIA (TensorRT, cuDNN)
  - ARM® (ARM Compute Library)

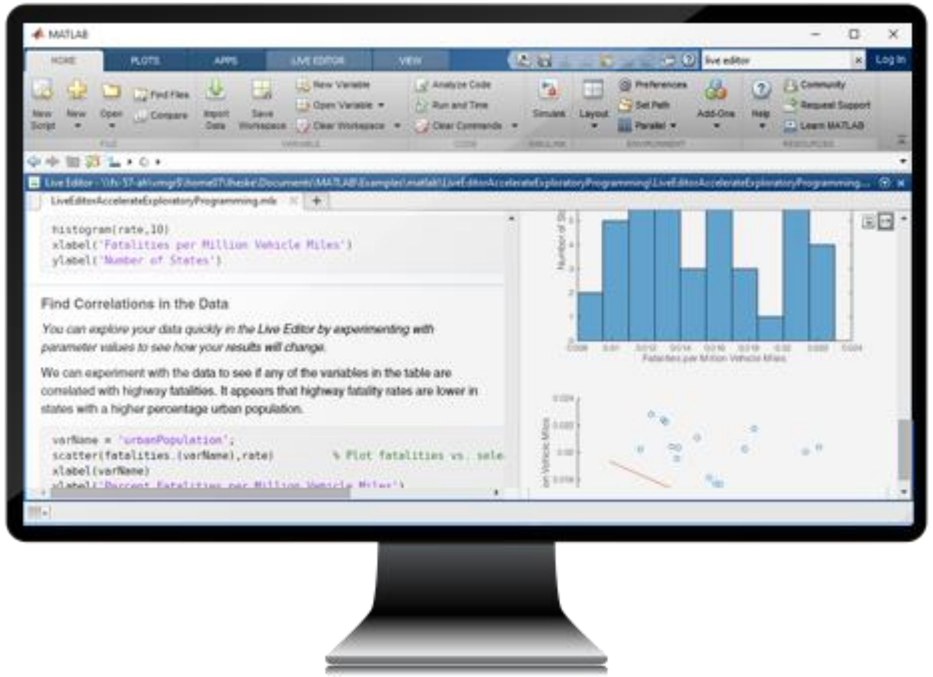
⇨ deployable models with high-performance inference speed.

Latest Features

- **What’s New in MATLAB for Deep Learning?**
  - https://www.mathworks.com/solutions/deep-learning/features.html

Jak začít s prostředím MATLAB?

- On-line kurzy zdarma
  - MATLAB Onramp, Simulink Onramp, Stateflow Onramp
  - Deep Learning Onramp, Machine Learning Onramp
  - časová náročnost: 2 hodiny
  - https://matlabacademy.mathworks.com/



```
histogram(rate,10)
xlabel('Fatalities per Million Vehicle Miles')
ylabel('Number of States')
```

Find Correlations in the Data

You can explore your data quickly in the Live Editor by experimenting with parameter values to see how your results will change.

We can experiment with the data to see if any of the variables in the table are correlated with highway fatalities. It appears that highway fatality rates are lower in states with a higher percentage urban population.

```
varName = 'urbanPopulation';
scatter(fatalities.(varName),rate)         % Plot fatalities vs. sele
xlabel(varName)
ylabel('Percent Fatalities per Million Vehicle Miles')
```

Děkuji za pozornost