Machine Learning with MATLAB

CME 192 LECTURE 5

02/05/2026

Outline

Statistical and Machine Learning

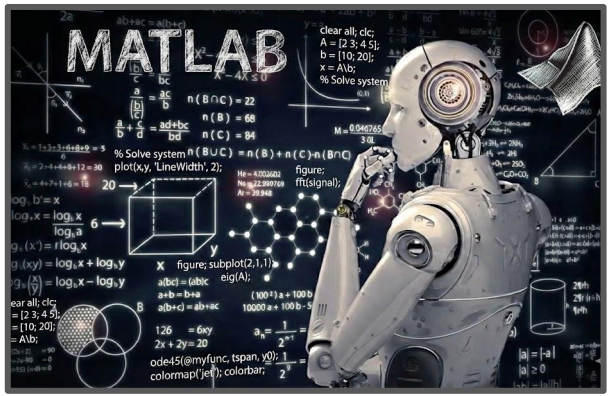
- ML refresher
- Data preprocessing and feature extraction
- Unsupervised learning
- Classification
- Regression

Deep Learning

- DL refresher
- Visualization methods
- Use pretrained model
- Create a network

Statistical and Machine Learning

How can we have machines learn patterns from data?

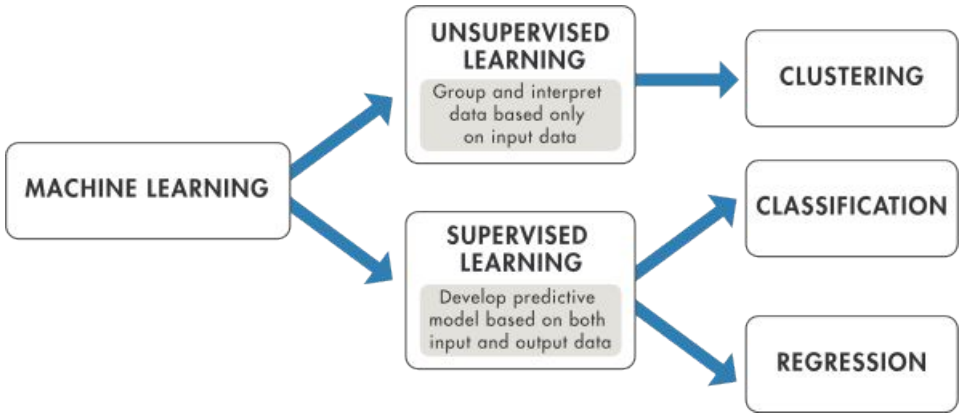


ML Refresher

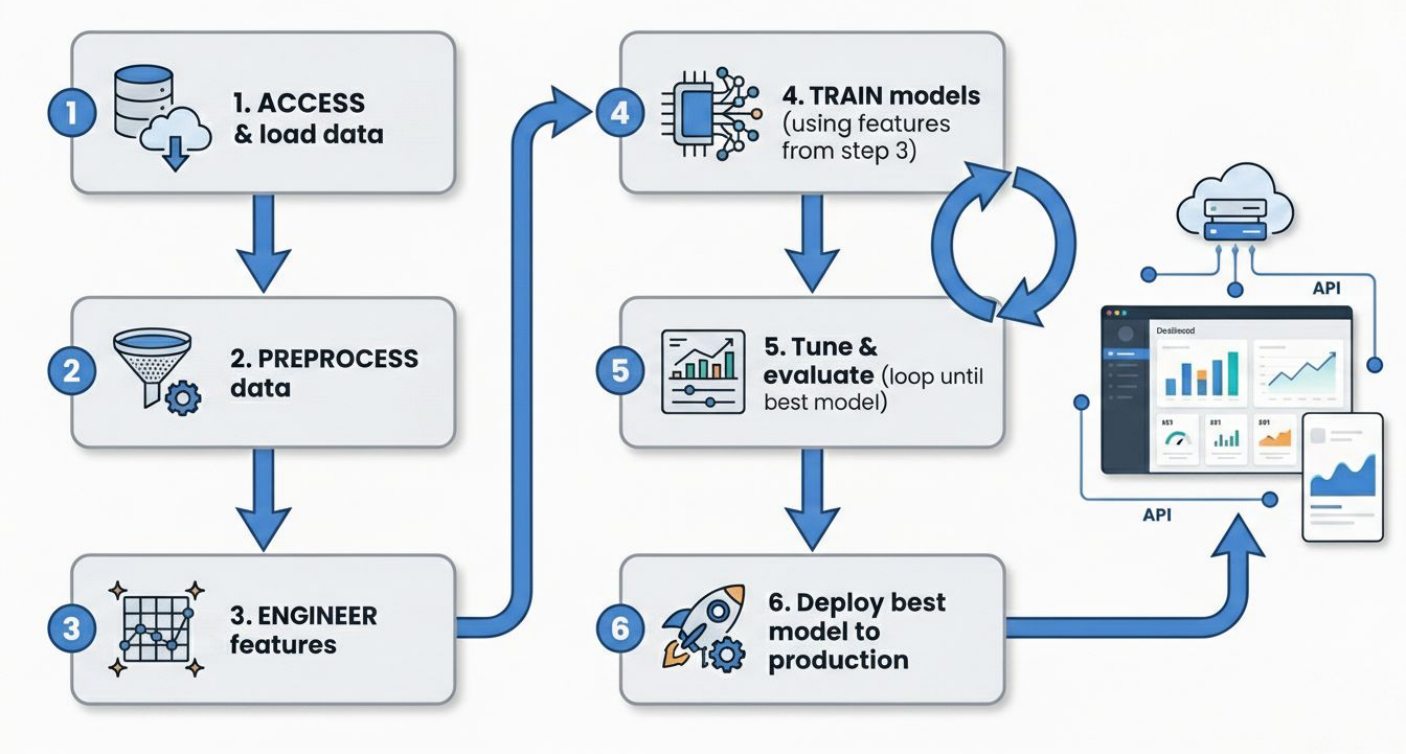
What is Machine Learning?

- Machine learning teaches computers to do what comes naturally to humans: learn from experience.
- ML algorithms use computational methods to “learn” information directly from data without relying on a predetermined equation as a model.
- The algorithms adaptively improve their performance as the number of samples available for learning increases.

Two types of techniques:



Systematic Machine Learning Workflow



Statistical and Machine Learning Toolbox

Statistics and Machine Learning Toolbox™ provides functions and apps to describe, analyze, and model data.

It provides tools and algorithms for:

- Descriptive statistics, visualizations, and clustering for exploratory data analysis
- PCA, regularization, dimensionality reduction, and feature selection methods for multidimensional data analysis and feature extraction
- Supervised, semi-supervised, and unsupervised methods

Install Toolbox:

Add-Ons • Get Add-Ons • Search for the toolbox • Install

Data preprocessing & feature extraction

Import and process Data

- `readtable` creates a table in MATLAB from a datafile
- `categorical` creates a categorical array from data.
- Assigning the empty array removes rows or columns. rmmissing removes any row with missing or undefined elements.
- `groupsummary` calculates statistics grouped according to a grouping variable.
- `innerjoin` merges two tables, retaining only the common key variable observations.

Visualize and engineer Features

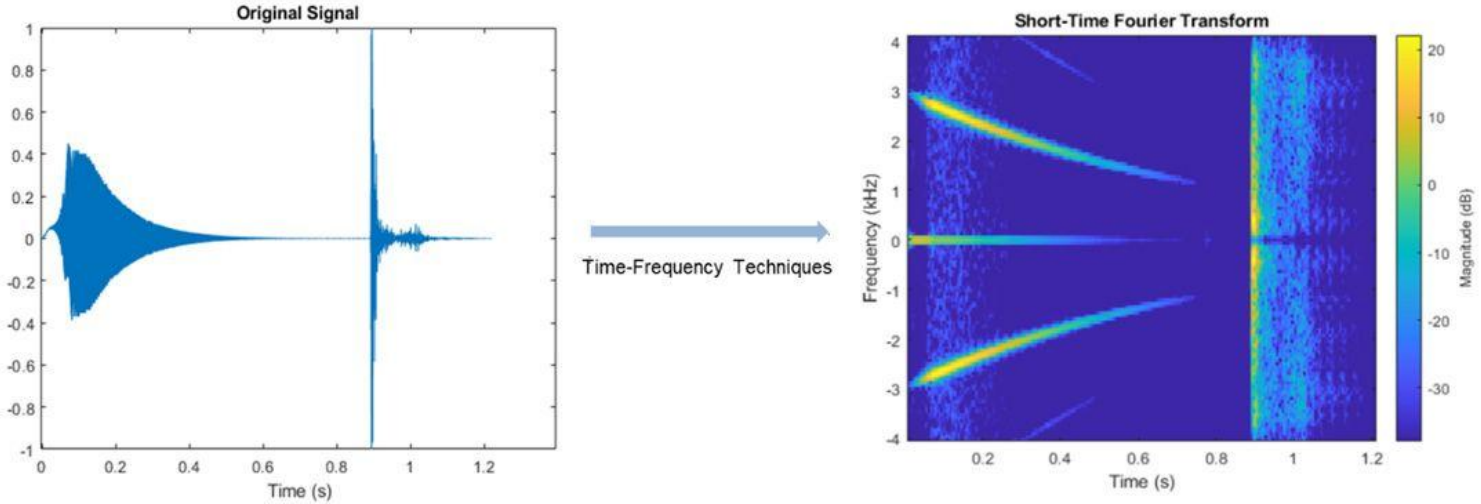
- `boxplot` can create separate box plots based on a grouping variable.
- Indexing and element-wise division can be used to scale variables in a table.
- `gscatter` creates a grouped scatter plot.

Calculate summary statistics

- Measures of Central Tendency
  - mean Arithmetic mean
  - median Median (middle) value
  - mode Most frequent value
  - trimmean Trimmed mean (mean, excluding outliers)
  - geomean Geometric mean
  - harmean Harmonic mean
- Measures of Spread
  - range Range of values (largest – smallest)
  - std Standard deviation
  - var Variance
  - mad Mean absolute deviation
  - iqr Interquartile range (75th percentile minus 25th percentile)
- Measures of Shape
  - skewness Skewness (third central moment)
  - kurtosis Kurtosis (fourth central moment)
  - moment Central moment of arbitrary order
- Other useful functions:
  - Finding peaks: islocalmax, islocalmin
  - Correlation between signals: corr

What is feature extraction?

Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. It yields better results than applying machine learning directly to the raw data.



What is feature extraction?

- Feature extraction can be done manually or automatically
- Manual feature extraction
  - Identify and describe which features matter for the problem
  - Implement a method to compute/extract those features from the data
  - Domain/background knowledge helps choose useful features
  - Many established methods exist for images, signals, and text
  - Example: mean value over a sliding window in a signal
- Automated feature extraction
  - Uses specialized algorithms or deep networks to learn/extract features directly from raw data
  - Speeds up going from raw data → machine learning model development
  - Example: wavelet scattering
- Impact of deep learning
  - For image data, feature extraction is often handled by the first layers of deep networks
  - For signals/time-series, feature extraction is still a major upfront challenge and often requires significant expertise to build effective models

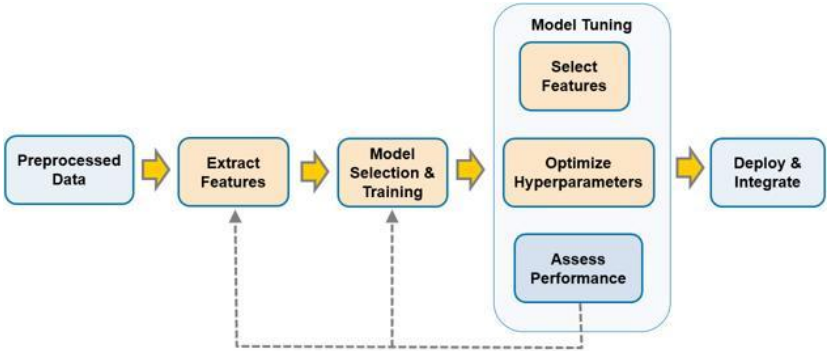
Automating Feature Extraction

One approach is to use custom processing (extraction) functions and transformed datastore

```
dsnew = transform (ds1, ds2, …, dsN, @fcn)
```

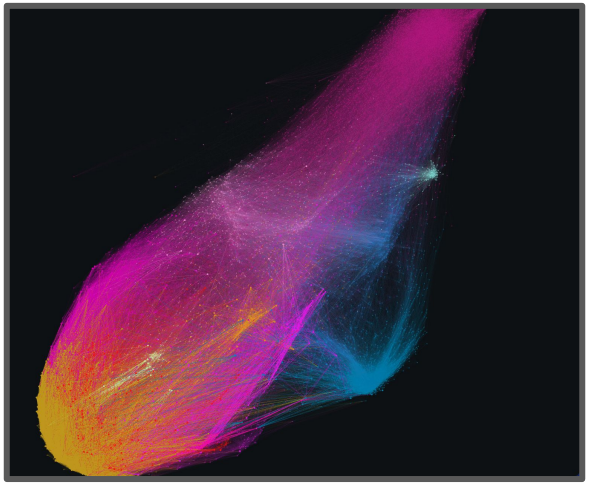
Where fcn is the transformation function.

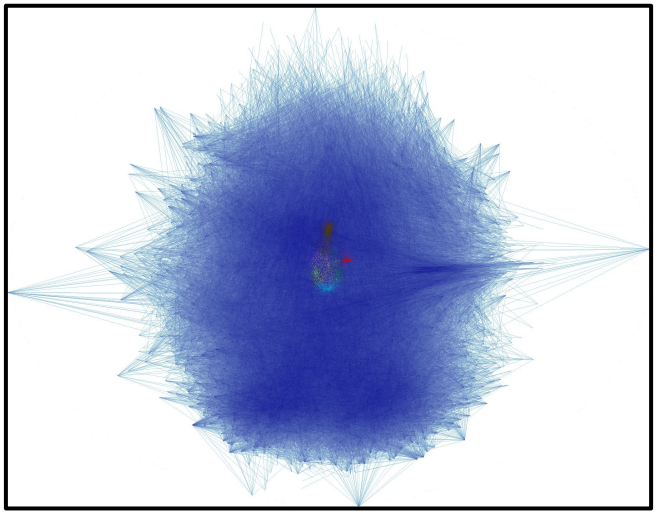
Another approach is thorough automated feature extraction



Unsupervised Learning

Finding patterns in data without labels (a specified objective)





Unsupervised learning

Goal: Discover hidden patterns or intrinsic structure in data

Setting: Learn from datasets consisting of input data without labeled responses

Two broad classes of unsupervised learning

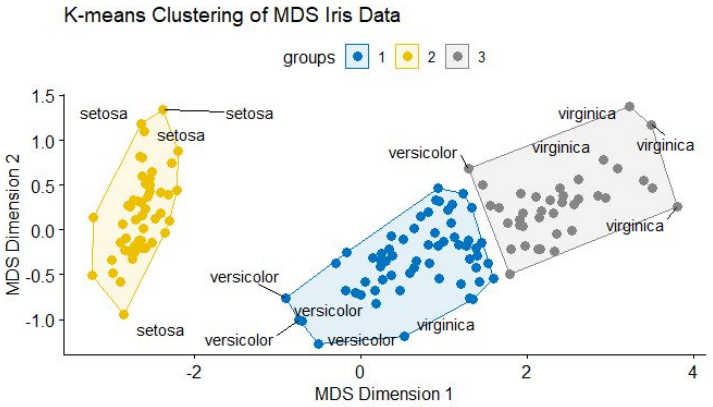
- **Clustering**: Identify groups of similar data points based on a chosen notion of similarity (e.g., k-means, hierarchical clustering, spectral clustering)
- **Dimensionality reduction**: Find low-dimensional representations that preserve important structure (e.g., PCA, autoencoders, manifold learning)

Note: Self-supervised learning is sometimes described as unsupervised, but it introduces artificial labels via a pretext task and is more accurately viewed as a separate paradigm.

Dimensionality Reduction

(Classical) Multidimensional Scaling (MDS, or CMD)

- Calculate pairwise distances `D = pdist(data, "distance")`
- Perform multidimensional scaling `[x, e] = cmdscale(D)`
  - X m-by-q matrix of the reconstructed coordinates in a q-dimensional space. q is the minimum number of dimensions to achieve the given pairwise distances.
  - e Eigenvalues of the matrix x * x'

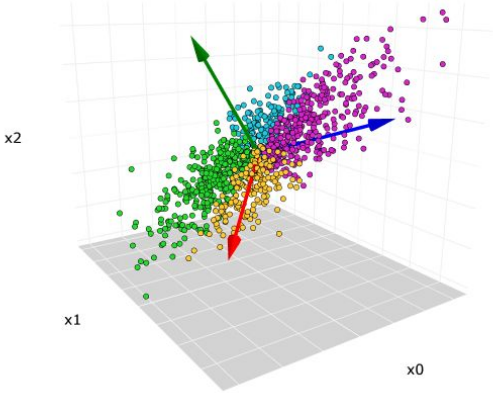


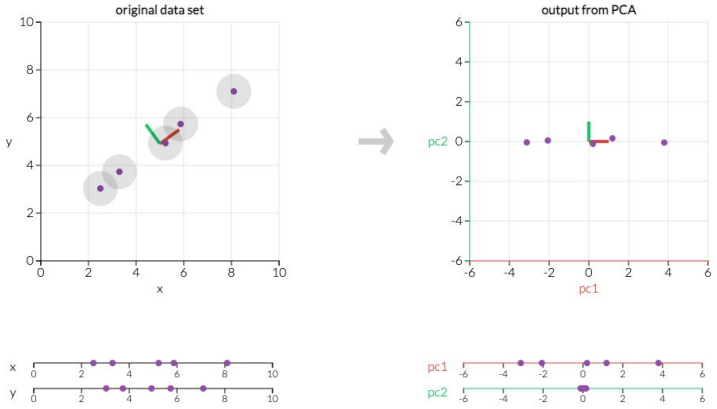
Dimensionality Reduction

Principal Component Analysis (PCA):
https://setosa.io/ev/principal-component-analysis/

- Perform `PCA [pcs,scrs,~,~,pexp] = pca(data)`
- `pcs` A n-by-n matrix of principal components.
- `scrs` An m-by-n matrix containing the data transformed using the linear coordinate transformation matrix pcs(first output).
- `pexp` A vector of length n containing the percentage of variance explained by each principal component.

CMD scaling is the same as PCA when using the 2-norm as the distance metric (within a potential minus sign).





Dimensionality Reduction

t-SNE (nonlinear, visualization-first): `Y = tsne(data,'NumDimensions',2)` for 2D/3D embeddings that often separate clusters well.

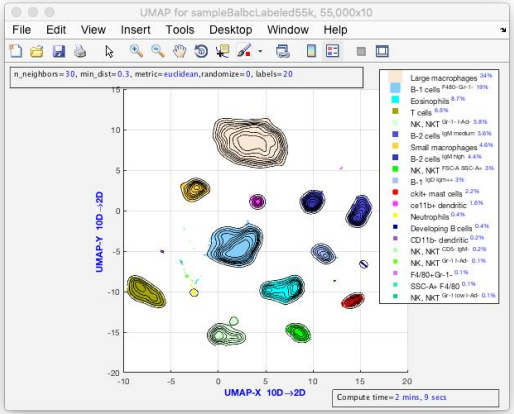
UMAP (nonlinear, fast + structure-preserving): commonly via an Add-On / File Exchange implementation (often called `run_umap`) for scalable 2D/3D manifold embeddings.

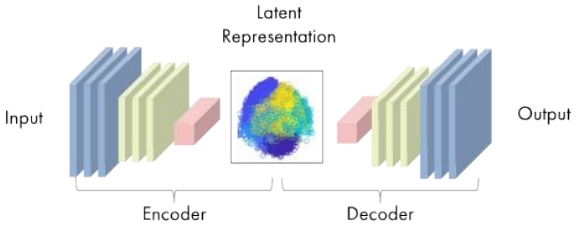
Autoencoders / deep embeddings (nonlinear, reconstructive): `autoenc = trainAutoencoder(data',h)` then use the bottleneck representation (and reconstruction) for compression.

Nonnegative Matrix Factorization (parts-based): `[W,H] = nnmf(data,k)` for interpretable low-dimensional factors when features are nonnegative.

RICA / ICA-style representations (independence-seeking): `M = rica(data,k)` then transform to latent features with transform(M,data).

Truncated SVD for large/sparse matrices (scalable linear baseline): `[U,S,V] = svds(X,k)` for low-rank approximation without needing the full SVD (useful when X is huge or sparse).





Dimensionality Reduction: Quick Example

```
rng(0)
m=4000; n=300; k=15;
A=rand(m,k)*rand(k,n)+1e-4*rand(m,n;
disp(size(A))

[coeff,score,~,~,~,mu]=pca(A);
Z=score(:,1:k);
disp(size(Z))
Ahat=Z*coeff(:,1:k)'+repmat(mu,m,1);
err_pca=norm(A-Ahat,'fro')/norm(A,'fro');
disp(err_pca)

Ac=A-repmat(mu,m,1);
[U,S,V]=svds(Ac,k);
disp(size(U))
Ahat2=U*S*V'+repmat(mu,m,1);
err_svd=norm(A-Ahat2,'fro')/norm(A,'fro');
disp(err_svd)
```

```
    4000         300

    4000          15

  7.3240e-06

    4000          15

  7.3240e-06

    4000          15

    0.0242
```

Clustering Algorithms

k-means Clustering `idx = kmeans(X,k)`

Partition the data into a fixed number of clusters by iteratively assigning each observation to the nearest cluster centroid and updating the centroids to minimize within-cluster variance.

- X Data, specified as a numeric matrix.
- k Number of clusters.
- idx Cluster indices, returned as a numeric column vector.
- optional inputs: "Distance"-Distance Metric, "Start"-Starting Locations of Cluster Centroids, "Replicates"-Replicates.

Gaussian Mixture Models (GMM)

Fit several n-dimensional normal distributions to the data and use those distributions to assign each observation to a cluster

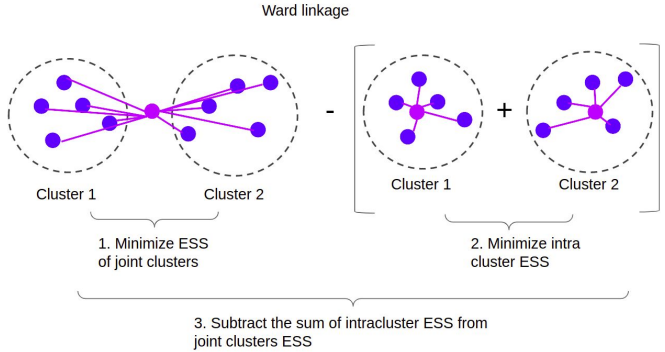
- Fit Gaussian Mixture Model: gm = fitgmdist(X,2);
- Identify Clusters: g = cluster(gm,X);[g,~,p] = cluster(gm,X);

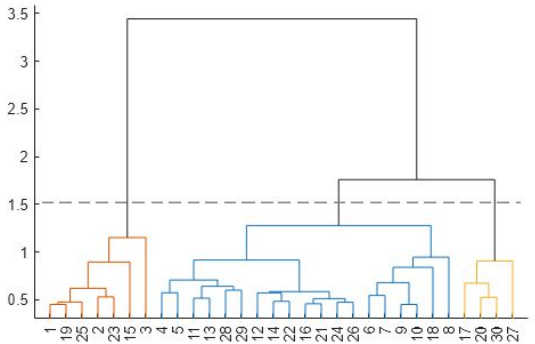
Hierarchical Clustering

Explore the sub-clusters that were grouped together to form bigger clusters.

- Determine Hierarchical Structure
  - use the linkage function to create the hierarchical tree.
  - use the dendrogram function to visualize the hierarchy.
- Divide Hierarchical Tree into Clusters
  - use the cluster function to assign observations into groups, according to the linkage distances Z.

```
Z = linkage(X,"centroid","cosine");
dendrogram(Z)
grp = cluster(Z,"maxclust",3)
```

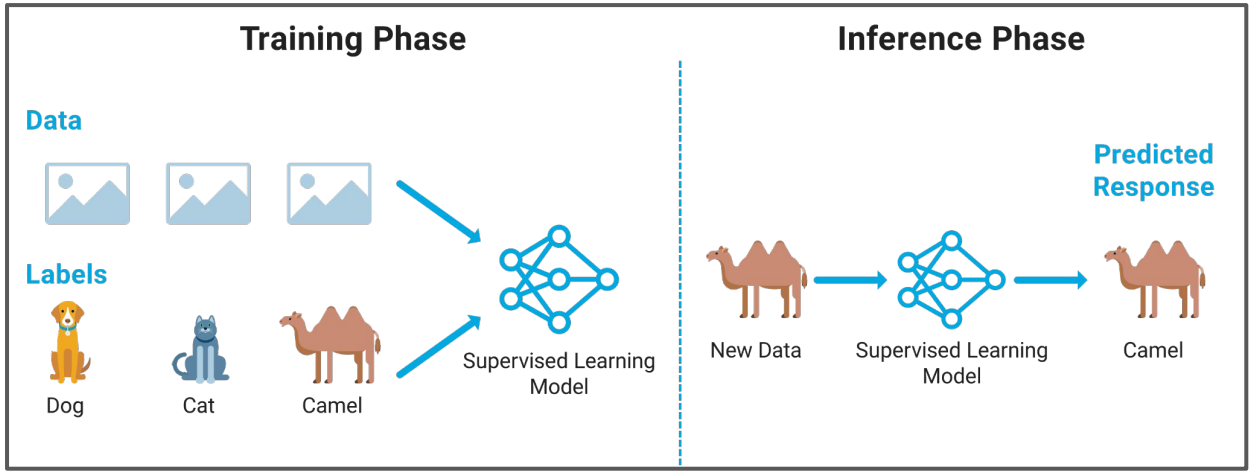




Interpret Clusters

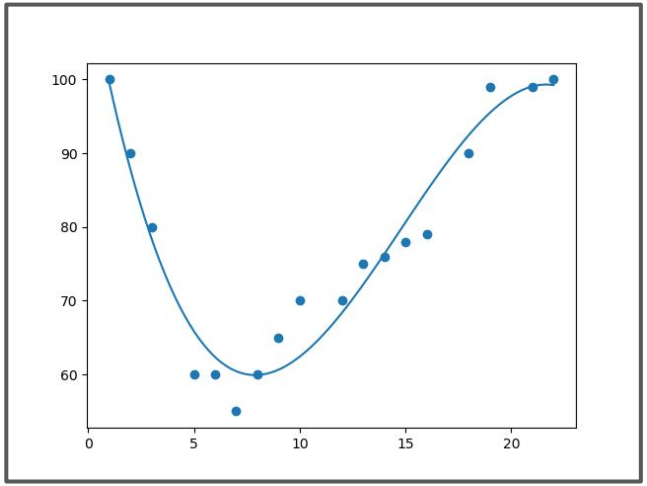
- Visualize group separation
- Create a silhouette plot
    - The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation).
    - [1, -1]
    - high value → the object is well matched to its own cluster and poorly matched to neighboring clusters

- Visualize cluster membership by attribute
- Create parallel coordinates plot of group mean
- Evaluate optimal number of clusters



Supervised Learning

Finding patterns in data following labels



Supervised Learning

The aim of supervised machine learning is to build a model that makes predictions based on evidence in the presence of uncertainty.

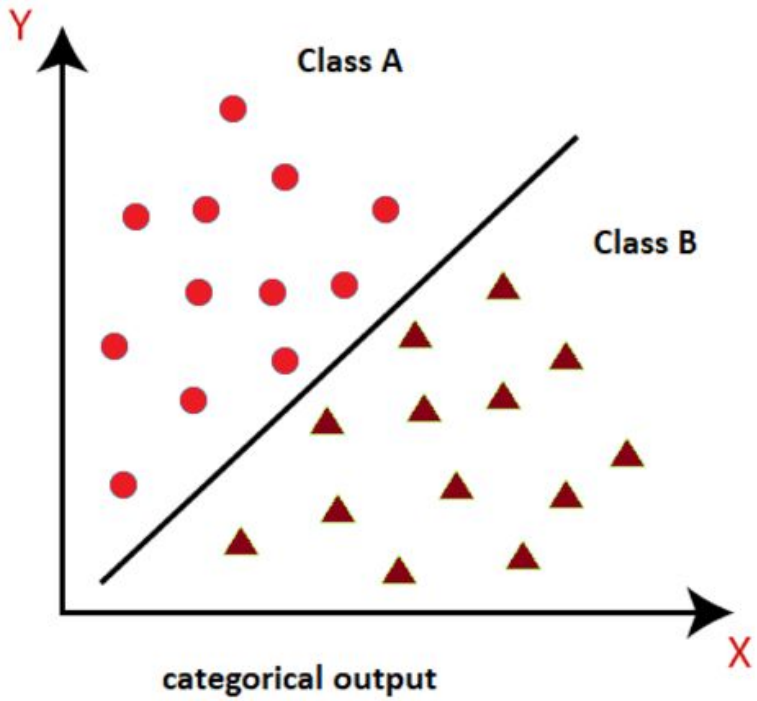
A **supervised learning algorithm** takes a known set of input data and known responses to the data (output) and trains a model to generate reasonable predictions for the response to new data.

- Classification
- Regression

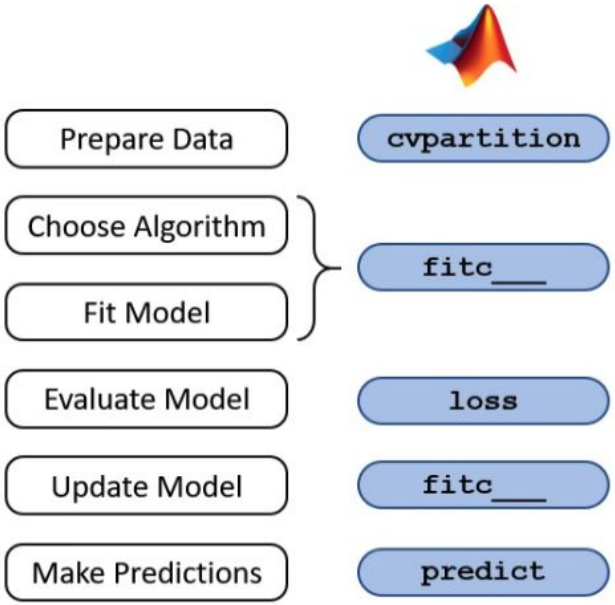
Classification

Classification techniques predict categorical responses, for example, whether an email is genuine or spam, or whether a tumor is cancerous or benign.

Typical applications:
medical imaging, image and speech recognition, credit scoring, …



Typical machine learning workflow



Model types

- decision trees
- discriminant analysis
- support vector machines
- logistic regression
- nearest neighbors
- naive Bayes
- kernel approximation
- ensemble
- neural network classifiers

Improving Predictive Models

- Cross Validation: Test how well a model will perform on new, unseen data by repeatedly training it on part of the data and evaluating it on the rest.

- Hyperparameter Optimization: Technique for choosing the best settings for a learning algorithm (like learning rate or model size) that are not learned directly from the data but strongly affect performance.

- Sequential Feature Selection: Systematically build a good set of input features by adding or removing features one at a time and keeping the changes that improve the model.

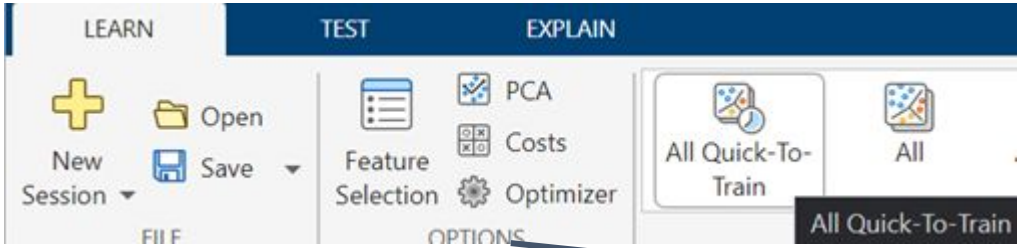
- Ensemble Learning: Multiple models are combined so that their collective predictions are more accurate/reliable than any single model alone.

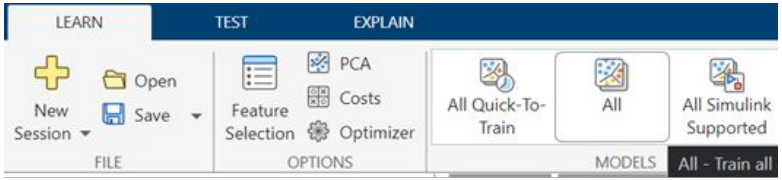
Interactive Tool: Classification Learner App

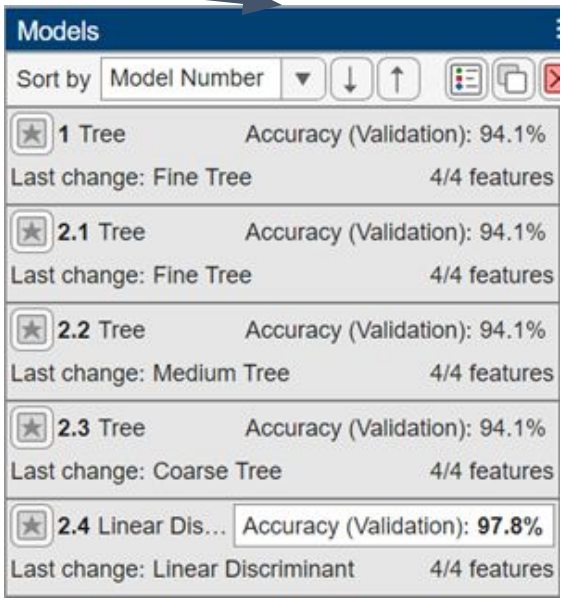
APPS → Classification Learner → New Session → Import data from workspace

- Purpose: Interactive app for supervised classification; train, compare, and analyze a wide range of classification models
- Explore the data, specify validation schemes, select features, and visualize results.
- Export models to the workspace to make predictions with new data.
- Generate MATLAB code from the app to create scripts, train with new data, work with huge data sets, or modify the code for further analysis.
- Workflow: Import labeled data (features + class labels) • Explore data and select predictors • Choose validation scheme (e.g., cross-validation) • Train models and optimize hyperparameters • Compare models using metrics and diagnostic plots

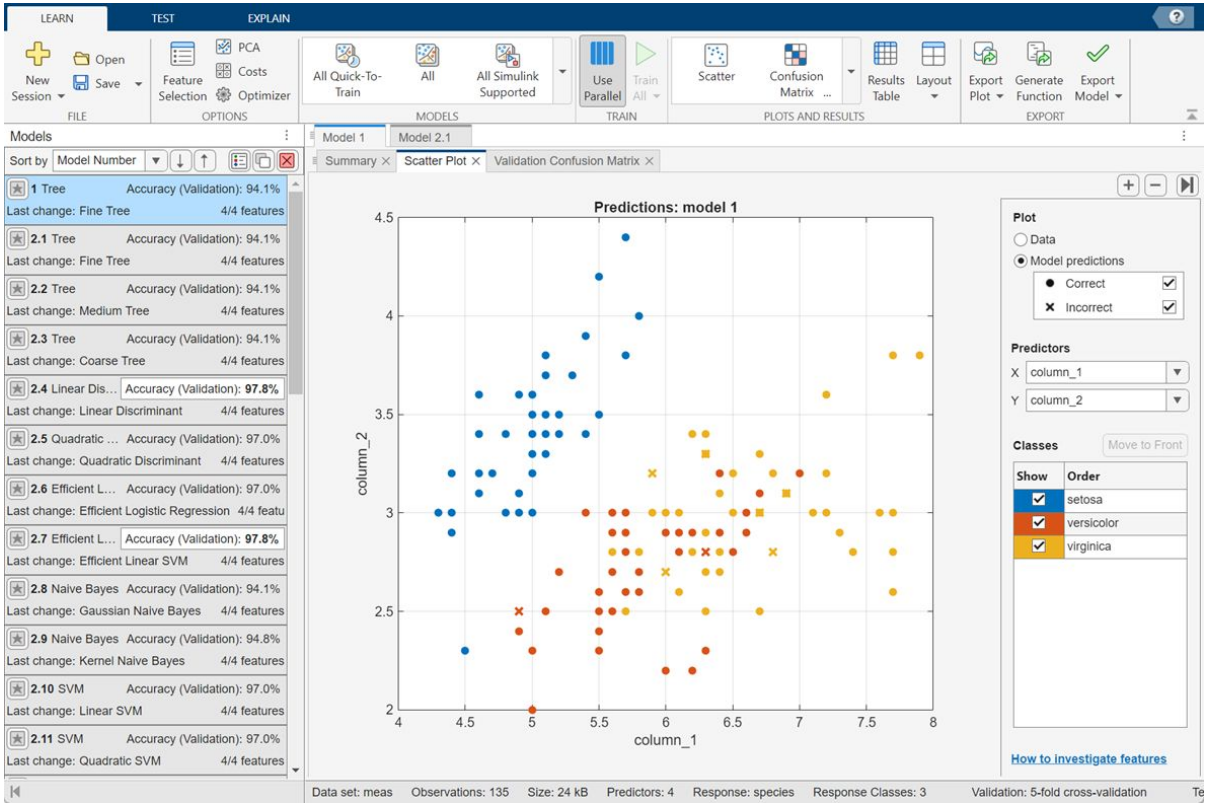
Classification Learner - ClassificationLearnerSession







Classification Learner - ClassificationLearnerSession



Regression

Regression techniques predict continuous responses, for example, changes in temperature or fluctuations in power demand.

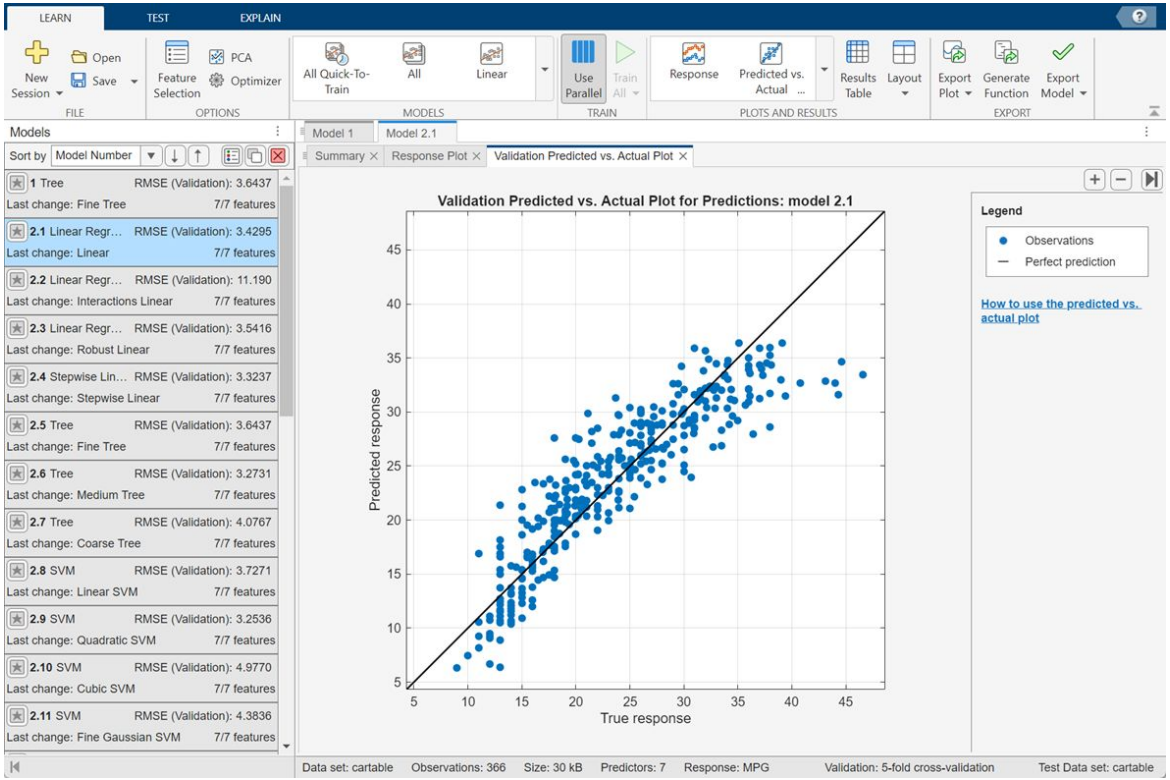
Typical applications: electricity load forecasting, algorithmic trading, …

Model types:

- linear regression models
- regression trees
- Gaussian process regression models
- support vector machines
- kernel approximation models
- ensembles of regression trees
- neural network regression models

Interactive Tool: Regression Learner App

** Functionalities similar to the Classification Learner App: "Choose among various algorithms to train and validate regression models. After training multiple models, compare their validation errors side-by-side, and then choose the best model."*



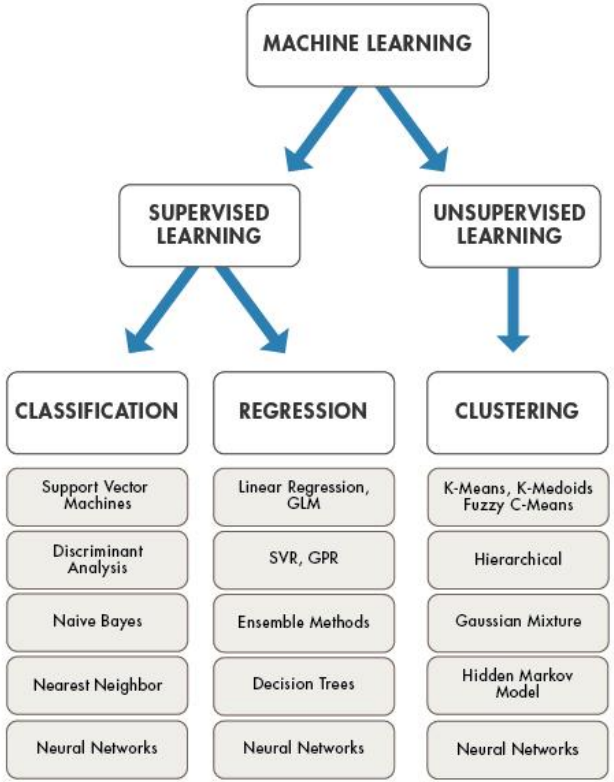
Choose the right algorithm

Tradeoff:

- Highly flexible models tend to overfit data by modeling minor variations that could be noise.
- Simple models are easier to interpret but might have lower accuracy.

Model speed, accuracy, and complexity

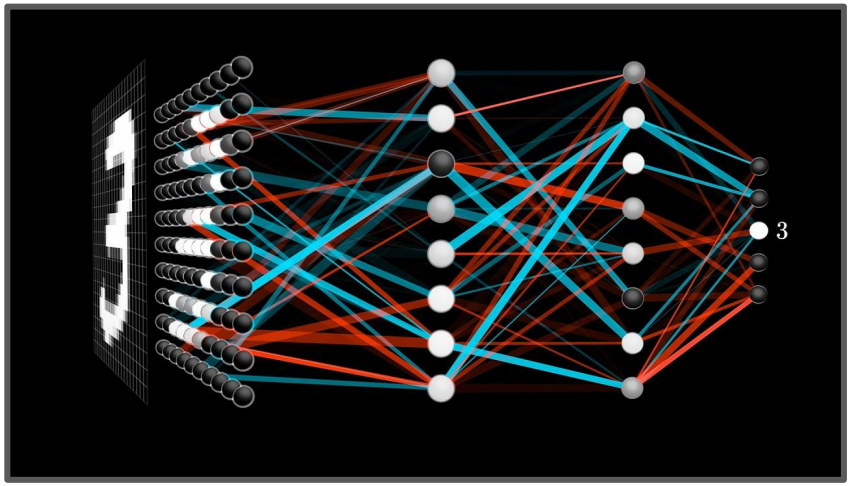
Trial and error

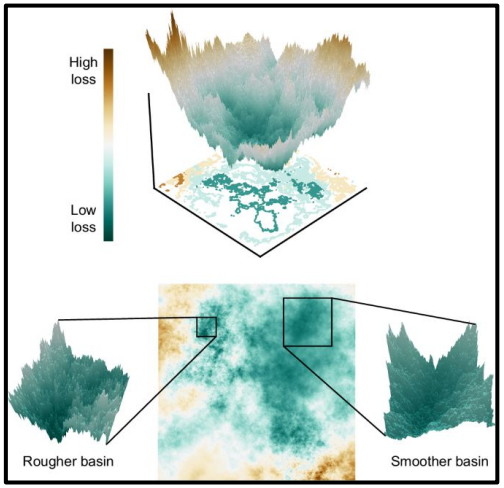


Task	MATLAB Apps and Functions	Product	Learn More
Classification to predict categorical responses	Use the Classification Learner app to automatically train a selection of models and help you choose the best. You can generate MATLAB code to work with scripts. For more options, you can use the command-line interface.	Statistics and Machine Learning Toolbox™	Train Classification Models in Classification Learner App Classification Functions
Regression to predict continuous responses	Use the Regression Learner app to automatically train a selection of models and help you choose the best. You can generate MATLAB code to work with scripts and other function options. For more options, you can use the command-line interface.	Statistics and Machine Learning Toolbox	Train Regression Models in Regression Learner App Regression Functions
Clustering	Use cluster analysis functions.	Statistics and Machine Learning Toolbox	Cluster Analysis and Anomaly Detection
Computational finance tasks such as credit scoring	Use tools for modeling credit risk analysis.	Financial Toolbox™ and Risk Management Toolbox™	Credit Risk (Financial Toolbox)
Deep learning with neural networks for classification and regression	Use pretrained networks and functions to train convolutional neural networks.	Deep Learning Toolbox™	Deep Learning in MATLAB (Deep Learning Toolbox)
Facial recognition, motion detection, and object detection	Use deep learning tools for image processing and computer vision.	Deep Learning Toolbox and Computer Vision Toolbox™	Recognition, Object Detection, and Semantic Segmentation (Computer Vision Toolbox)

Deep Learning

This is deep!

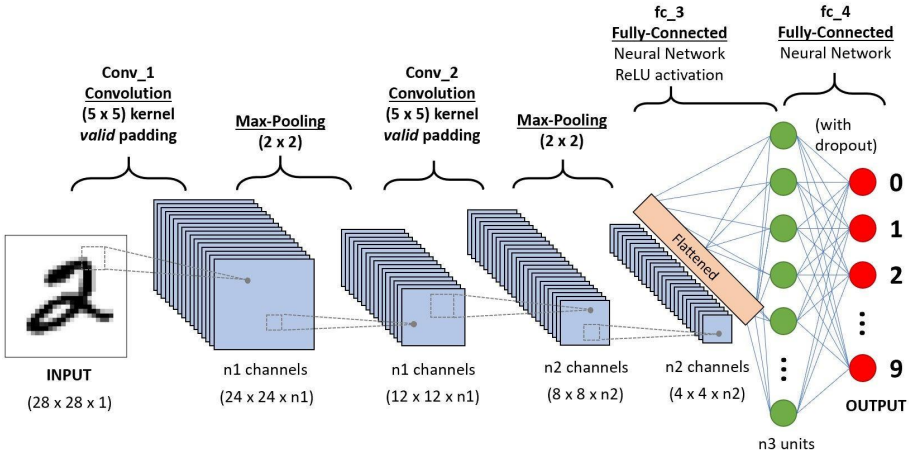




DL Refresher

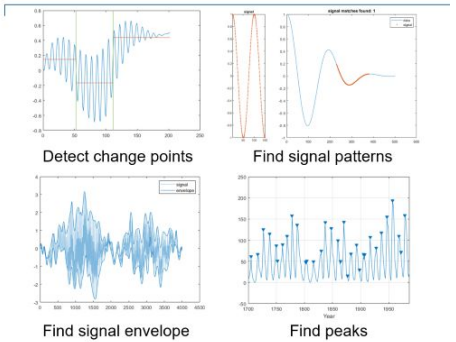
What Is Deep Learning?

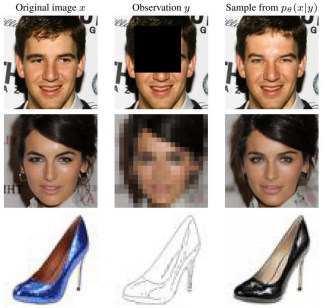
- Deep learning is a branch of machine learning that teaches computers to do what comes naturally to humans: learn from experience.
- DL uses neural networks to learn useful representations of features directly from data.
- NNs combine multiple nonlinear processing layers, using simple elements operating in parallel and inspired by biological nervous systems.

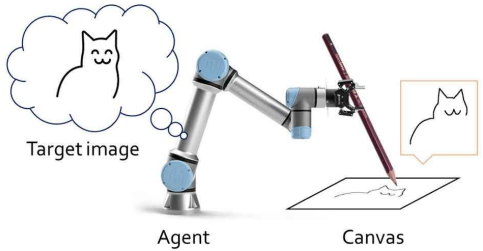


Deep learning applications

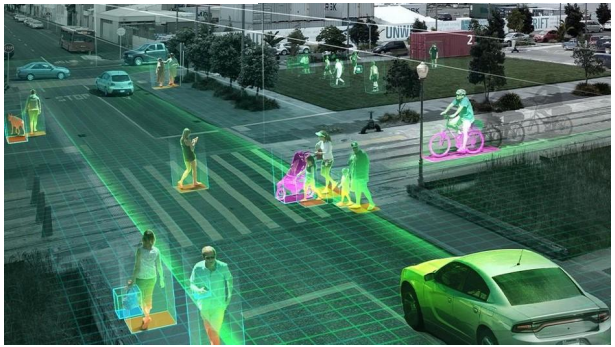
- Image classification, regression, and processing
- Sequences and time series
- Computer vision
- Audio processing
- Automated driving
- Signal processing
- Wireless communications
- Reinforcement learning
- Computational finance
- Lidar processing
- Text analytics
- Predictive maintenance

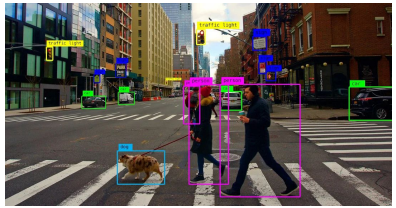






Once it sees the target image, it starts to draw what it saw on the canvas.

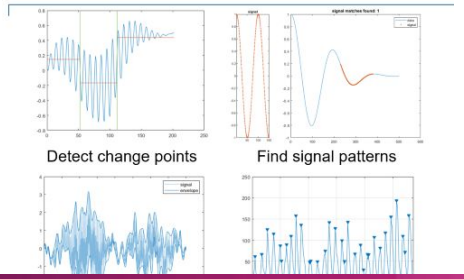


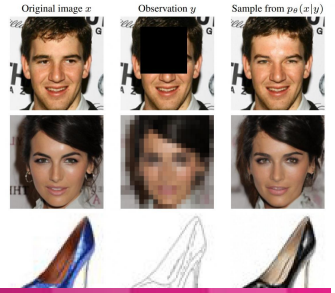


Stanford University

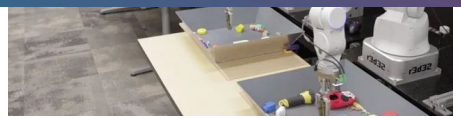
Deep learning applications

- Image classification, regression, and processing
- Sequences and time series
- Computer vision
- Audio processing
- Automated driving
- Signal processing
- Wireless communications
- Reinforcement learning
- Computational finance
- Lidar processing
- Text analytics
- Predictive maintenance





Once it sees



Deep Learning Toolbox

Deep Learning Toolbox™ provides a framework for designing and implementing deep neural networks with algorithms, pretrained models, and apps.

- use convolutional neural networks (ConvNets, CNNs) and long short-term memory (LSTM) networks to perform classification and regression on image, time-series, and text data
- build network architectures such as generative adversarial networks (GANs) and Siamese networks using automatic differentiation, custom training loops, and shared weights
- design, analyze, and train networks graphically with the Deep Network Designer app
- exchange models with TensorFlow and PyTorch through the ONNX format and import models from TensorFlow-Keras and Caffe

Use Pretrained Model: an example using GoogLeNet

GoogLeNet is a pretrained model that has been trained on a subset of the ImageNet database which is used in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC).

The model is trained on more than a million images, has 144 layers, and can classify images into 1000 object categories (e.g. keyboard, mouse, pencil, and many animals).

- Install the package use the Add-On Explorer.
- Load Pretrained Network
    - If the required support packages is not installed, then the software provides a download link

```
net = googlenet;
```

- Display layers of GoogLeNet

  `net.Layers`

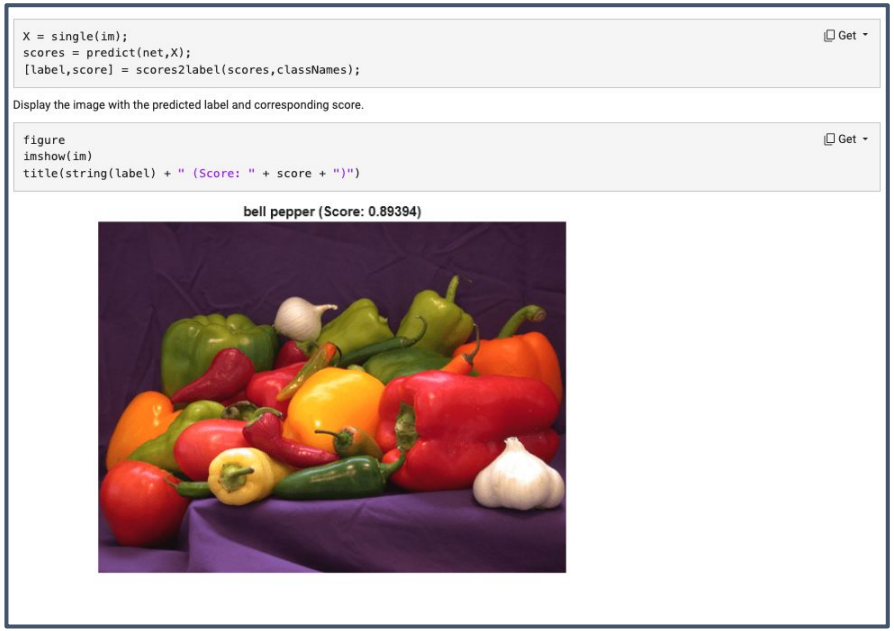
- View network architecture

  `analyzeNetwork(net)`

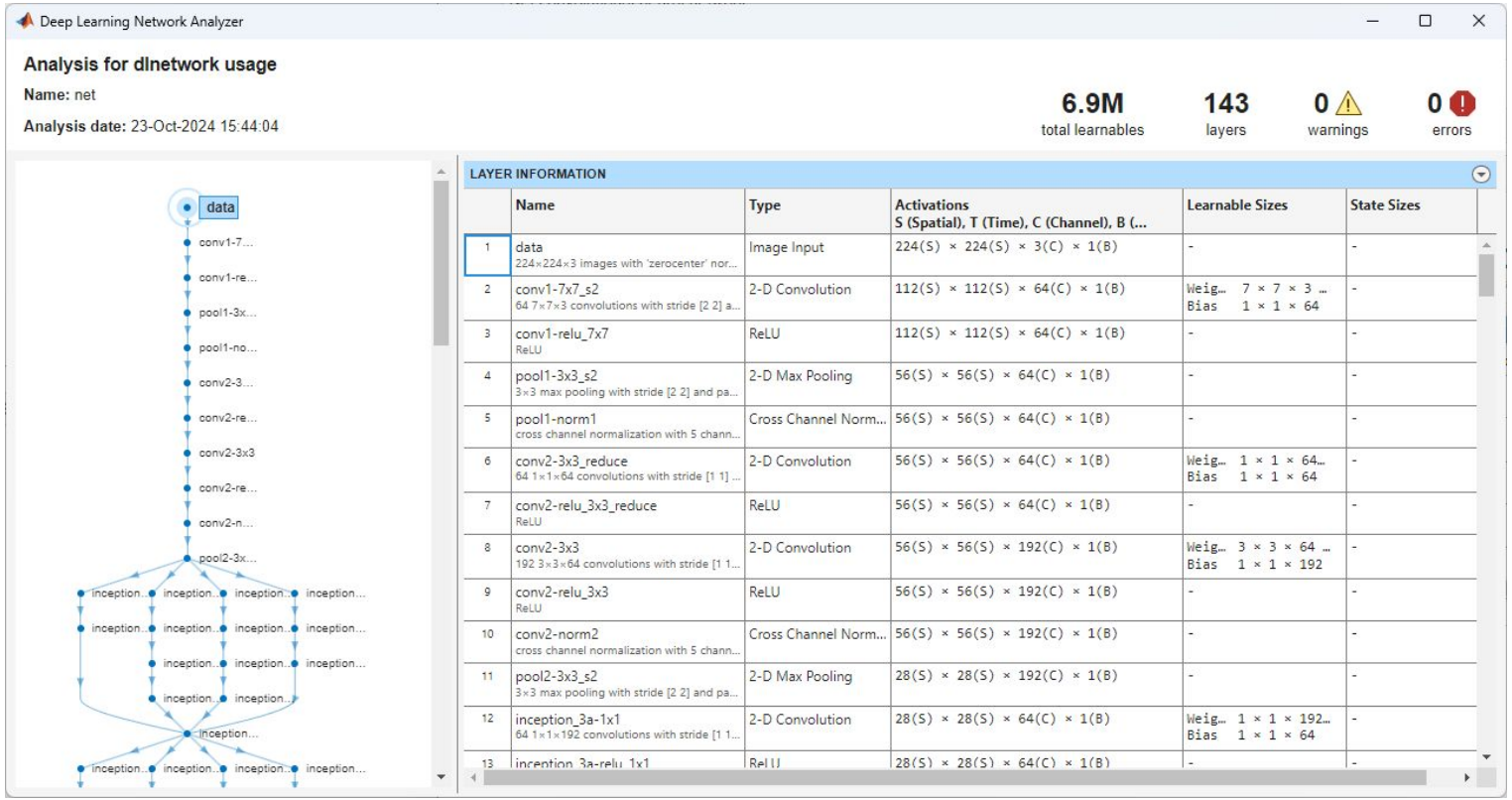
- Load an image and resize if needed

  `I = imread("image.png");`

- Classify the image and display the label.



Deep Learning Network Analyzer



Deep Learning Visualization Methods

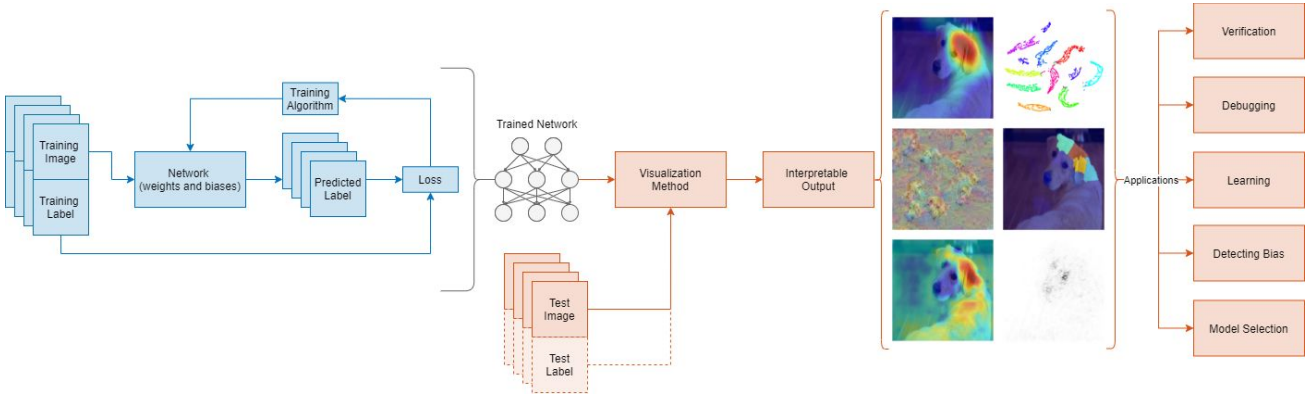
“Black Boxes”: understanding why a network makes a particular decision is crucial.

Interpretability techniques

- Translate network behavior into output that a person can interpret.
- Answer questions about the predictions of a network.
- Applications: verification, debugging, learning, assessing bias, and model selection.

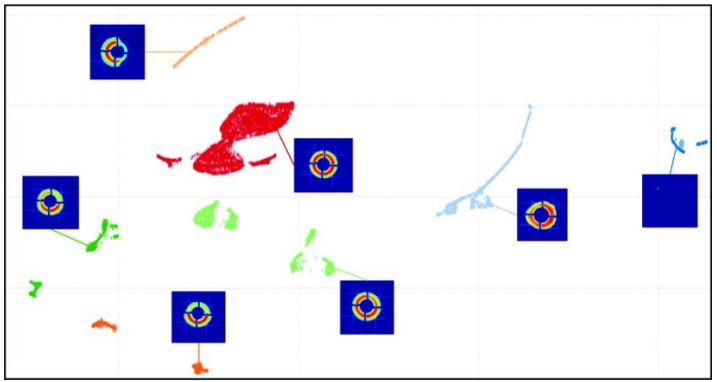
Visualization methods:

- Techniques: heat maps, saliency maps, feature importance maps, low-dimensional projections, …



Visualization interpretability techniques for image classification:

- Activation visualization
- CAM and Grad-CAM (gradient-based class activation heat map)
- Occlusion sensitivity (perturbation-based heat map)
- LIME (perturbation-based proxy model, feature)
- Gradient attribution (gradient-based saliency map)
- Deep dream (gradient-based activation maximization)
- t-SNE (dimension reduction)



E.g., extract features from an input image using a CNN with the activations function.

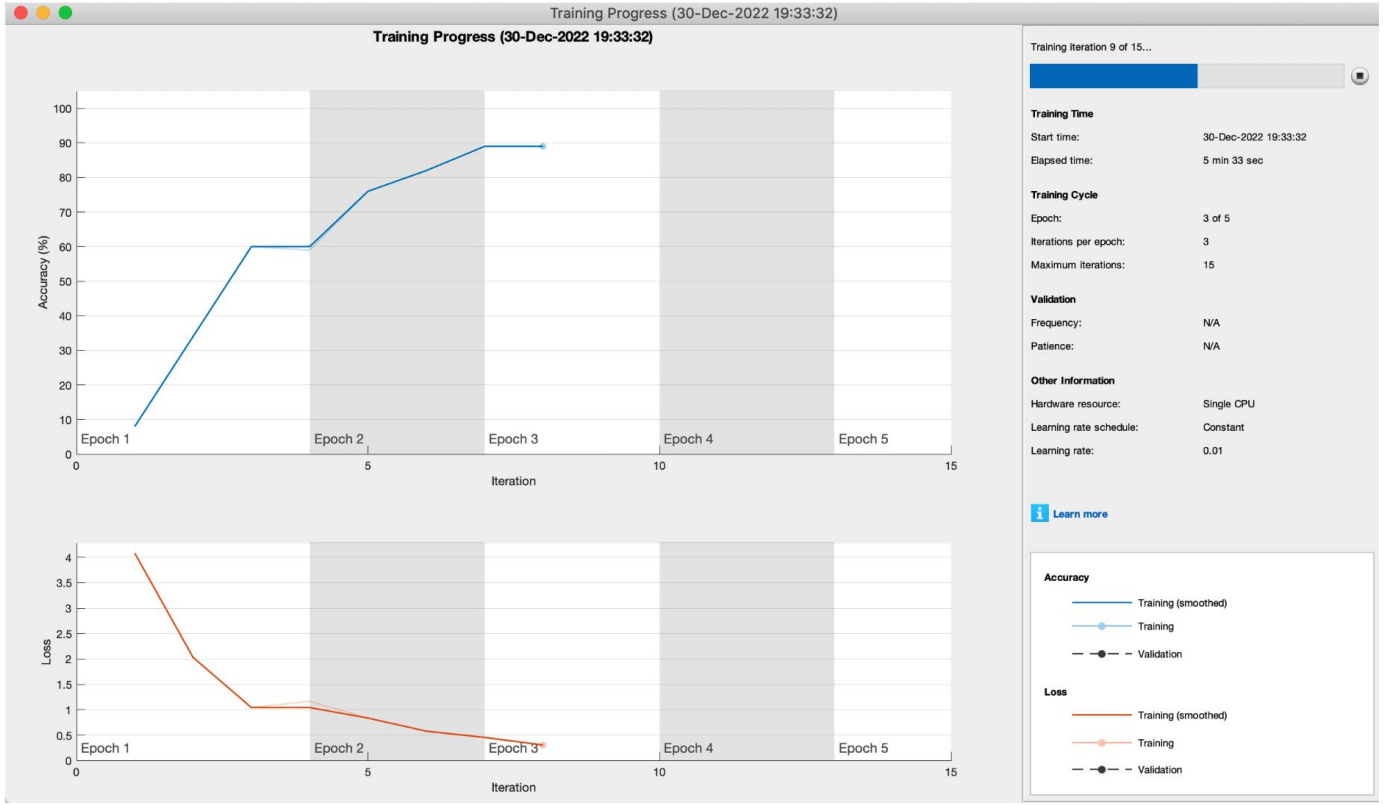
```
Y = activations(net, X, layerOut)
```

- It returns network activations for a specific layer using the network net and the data X. Network activations are computed by forward propagating the input X through the network up to the specified layer.

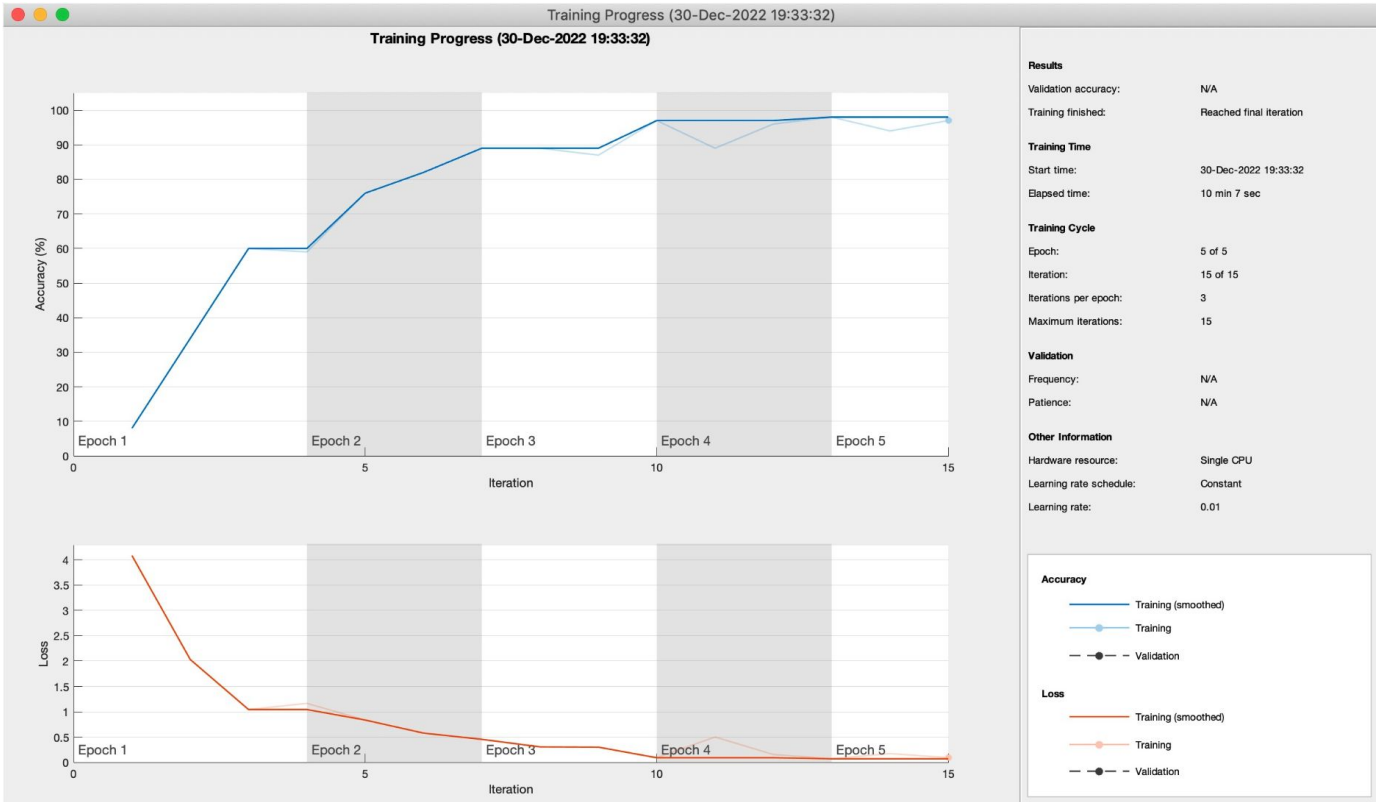
Create a network by modifying a pretrained model

1. Load pretrained model and modify layer(s).
2. Set training algorithm options.
   - An iteration is one step taken in the gradient descent algorithm towards minimizing the loss function using a mini-batch.
   - An epoch is the full pass of the training algorithm over the entire training set
   - Iterations per epoch = Number of training samples / Mini-batch size
   - Number of iterations = Iterations per epoch * Number of epochs
3. Perform training.
4. Use the trained network to make classification.
5. Evaluate the results.

Training Progress

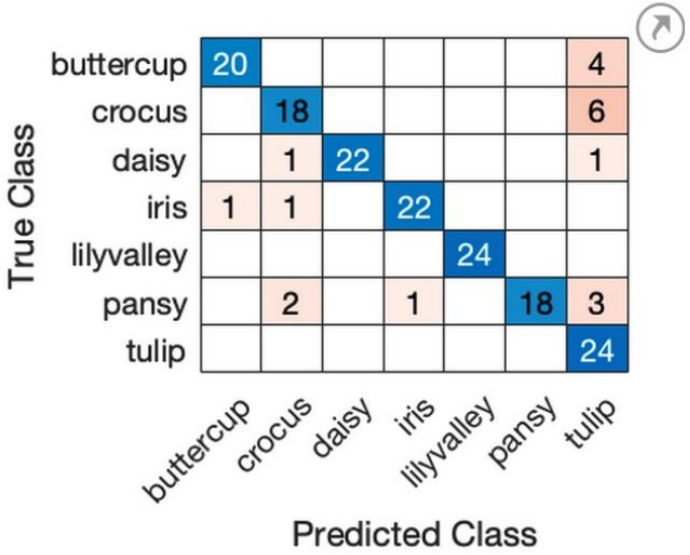


Training Progress



Accuracy: 0.881

Confusion chart:

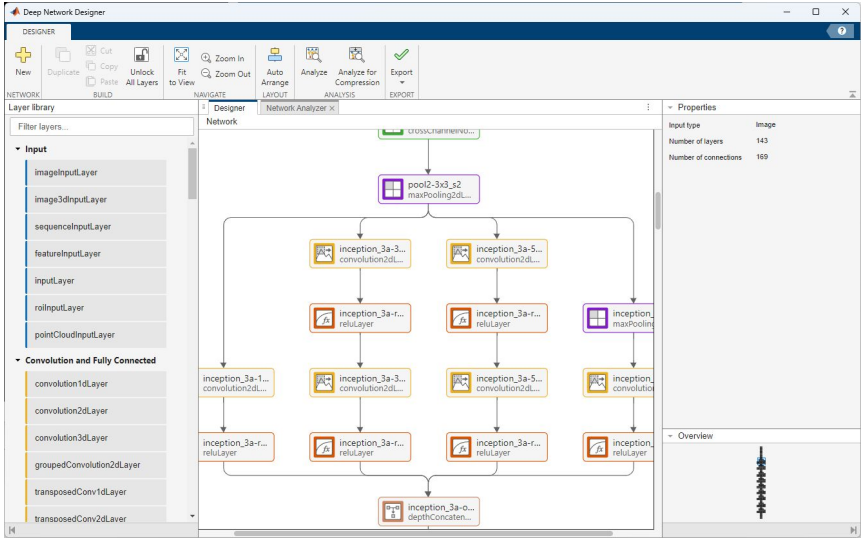


Interactive Deep Learning Apps

- Deep Network Designer
- Experiment Manager
- Deep Network Quantizer
- Reinforcement Learning Designer
- Image Labeler
- Video Labeler
- Ground Truth Labeler
- Lidar Labeler
- Signal Labeler

Deep Network Designer

- Build, import, edit, and combine networks.
- Load pretrained networks and edit for transfer learning.
- View and edit layer properties and add new layers and connections.
- Analyze the network and detect problems before training.
- Import and visualize datastores and image data for training and validation.
- Apply augmentations to image classification training data and visualize the distribution of the class labels.
- Train networks and monitor training with plots of accuracy, loss, and validation metrics.
- Export trained networks to the workspace or to Simulink.
- Generate MATLAB code for building and training networks and create experiments for hyperparameter tuning using Experiment Manager.

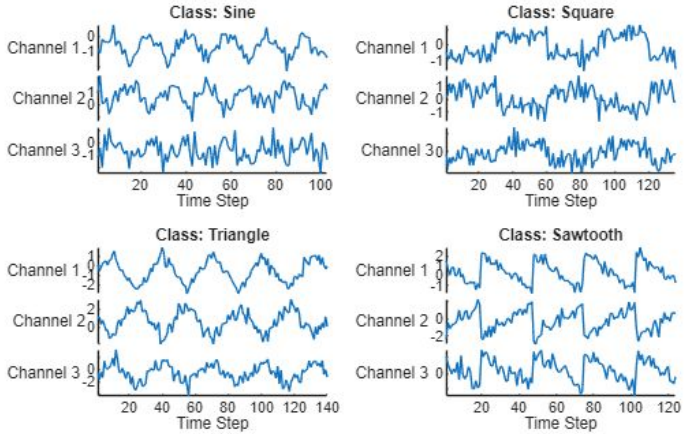


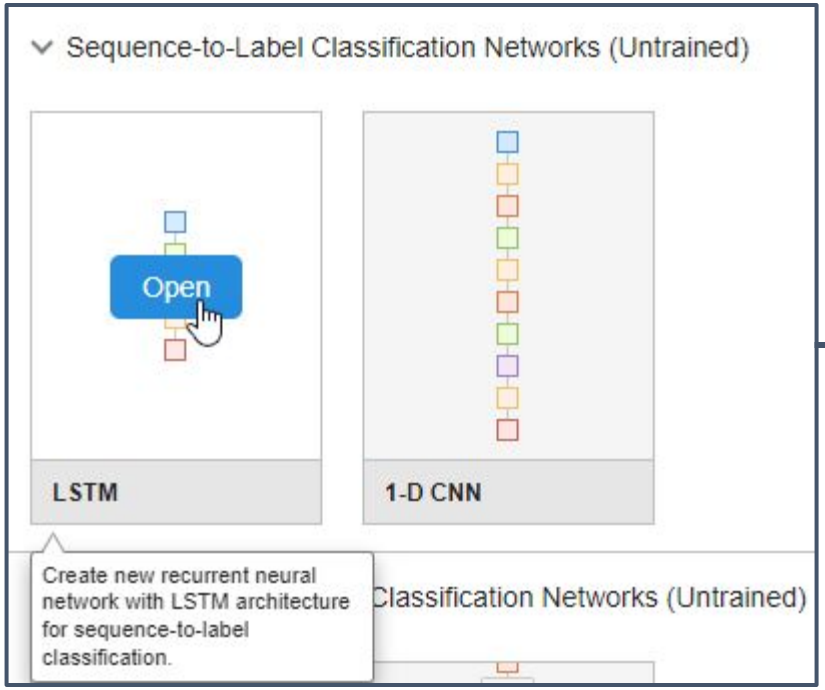
Open the Deep Network Designer App

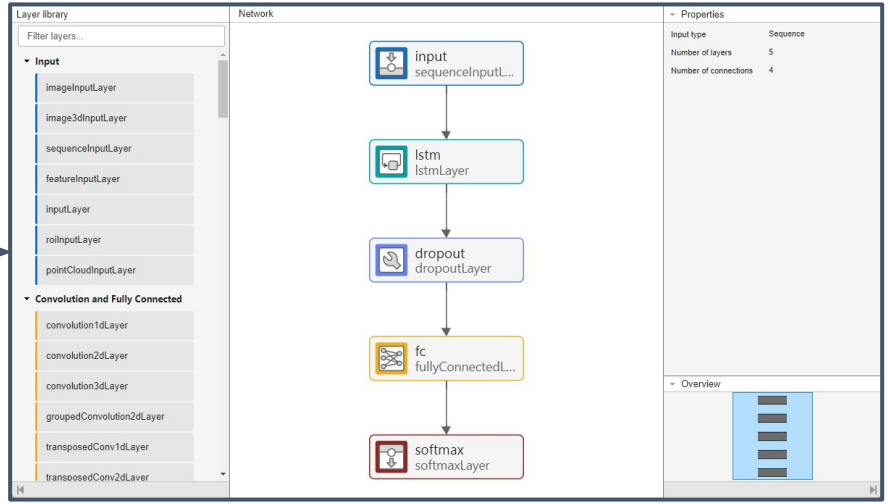
- MATLAB Toolstrip: Apps → Machine Learning and Deep Learning → Deep Network Designer.
- MATLAB command prompt: deepNetworkDesigner

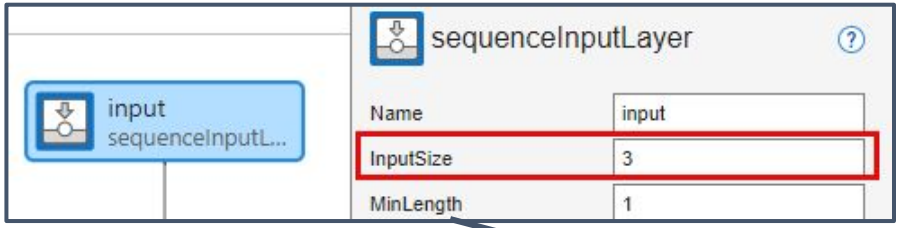
Create Network Architectures from Scratch

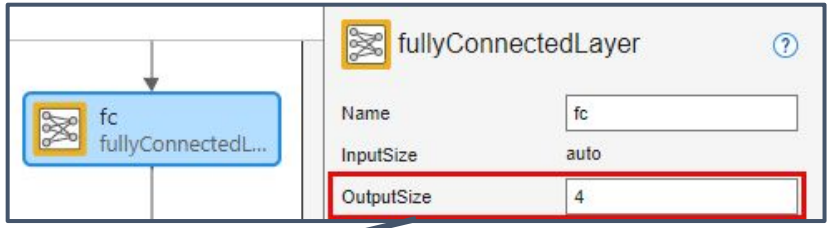
1. Load, process, and split data
2. Create architecture
3. Train the network
4. Make predictions
5. Evaluate the network




Sequence-to-Label Classification Networks (Untrained)
Open
LSTM
1-D CNN
Create new recurrent neural network with LSTM architecture for sequence-to-label classification.
Classification Networks (Untrained)


Layer library
Filter layers...
Input
imageInputLayer
image3dInputLayer
sequenceInputLayer
featureInputLayer
inputLayer
roiInputLayer
pointCloudInputLayer
Convolution and Fully Connected
convolution1dLayer
convolution2dLayer
convolution3dLayer
groupedConvolution2dLayer
transposedConv1dLayer
transposedConv2dLayer
Network
input
sequenceInputL...
lstm
lstmLayer
dropout
dropoutLayer
fc
fullyConnectedL...
softmax
softmaxLayer
Properties
Input type
Sequence
Number of layers
5
Number of connections
4
Overview





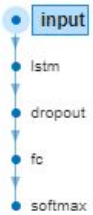
Designer | Network Analyzer

Analysis for dlnetwork usage

Name: Network from Deep Network Designer

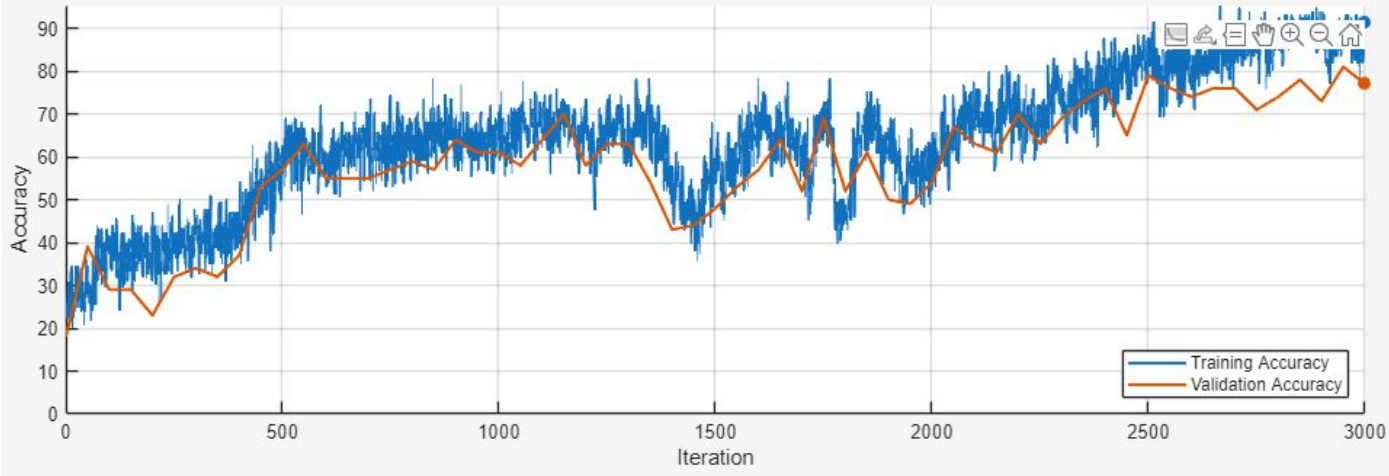
Analysis date: 06-Nov-2024 09:59:22

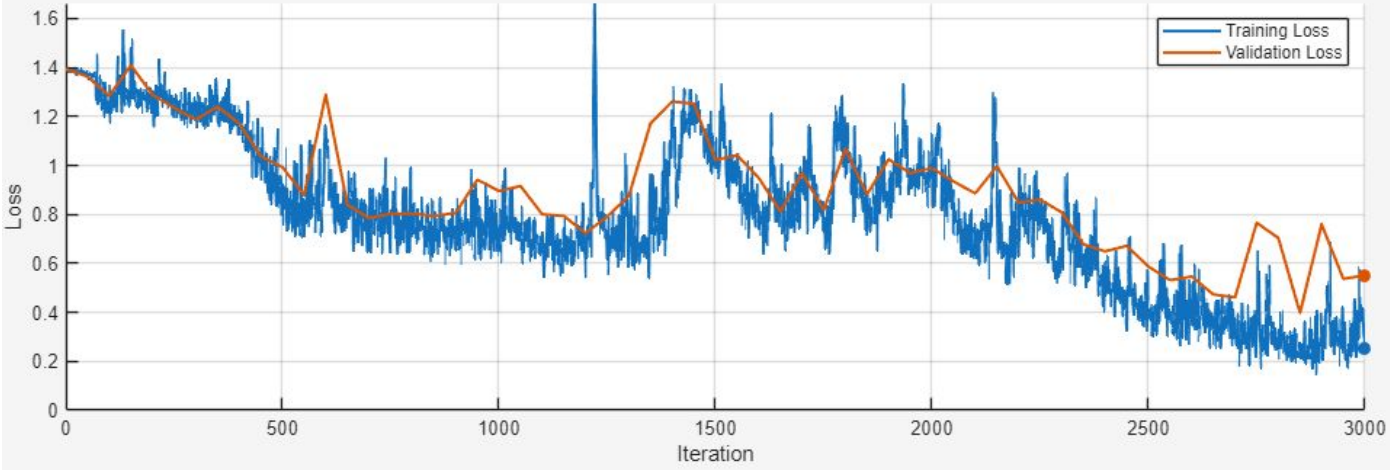
68.1k	5	0	0
total learnables	layers	warnings	errors



LAYER INFORMATION

	Name	Type	Activations S (Spatial), T (Time), C (Channel), B (...	Learnable Sizes	State Sizes
1	input Sequence input with 3 dimensions	Sequence Input	3(C) × 1(B) × 1(T)	-	-
2	lstm LSTM with 128 hidden units	LSTM	128(C) × 1(B)	InputWeigh… 512 ×… RecurrentW… 512 ×… Bias 512 ×…	HiddenSta… CellState
3	dropout 50% dropout	Dropout	128(C) × 1(B)	-	-
4	fc 4 fully connected layer	Fully Connected	4(C) × 1(B)	Weights 4 × 128 Bias 4 × 1	-
5	softmax softmax	Softmax	4(C) × 1(B)	-	-





Progress:

Status: Training stopped

Stop reason: Max epochs completed

Time

Start time:	06-Nov-2024 10:28:10
Elapsed time:	00:04:55

Information

Epoch:	500 of 500
Iteration:	3000
Learning rate schedule:	Constant
Learning rate:	0.0005
Validation frequency:	50
Validation patience:	Inf
Objective metric:	Loss
Output network:	Best validation
Hardware resource:	Single GPU

Export as Image

```
acc = mean(YTest == TTest)
```

Get

```
acc =
0.8000
```

Visualize the predictions in a confusion chart.

```
figure
confusionchart(TTest,YTest)
```

Get

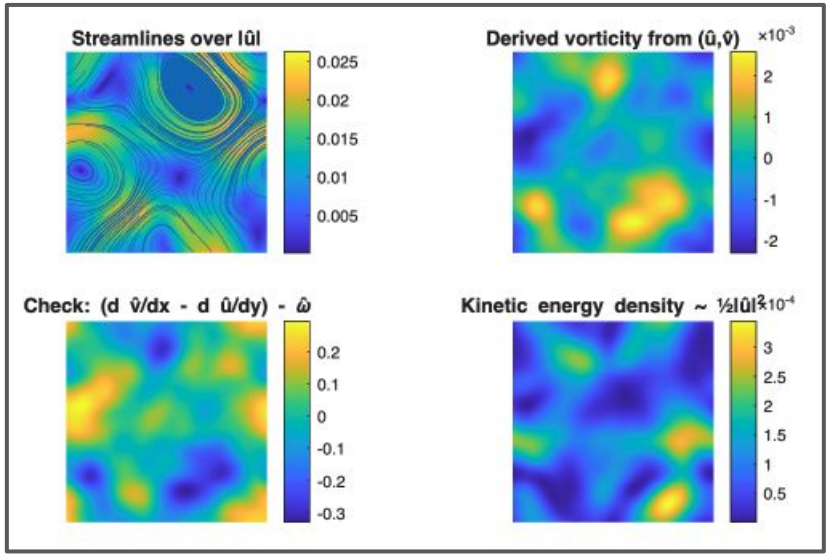
If you are interested in training a Generative Adversarial Network (GAN) in MATLAB, check out this page:
https://www.mathworks.com/discovery/generative-adversarial-networks.html

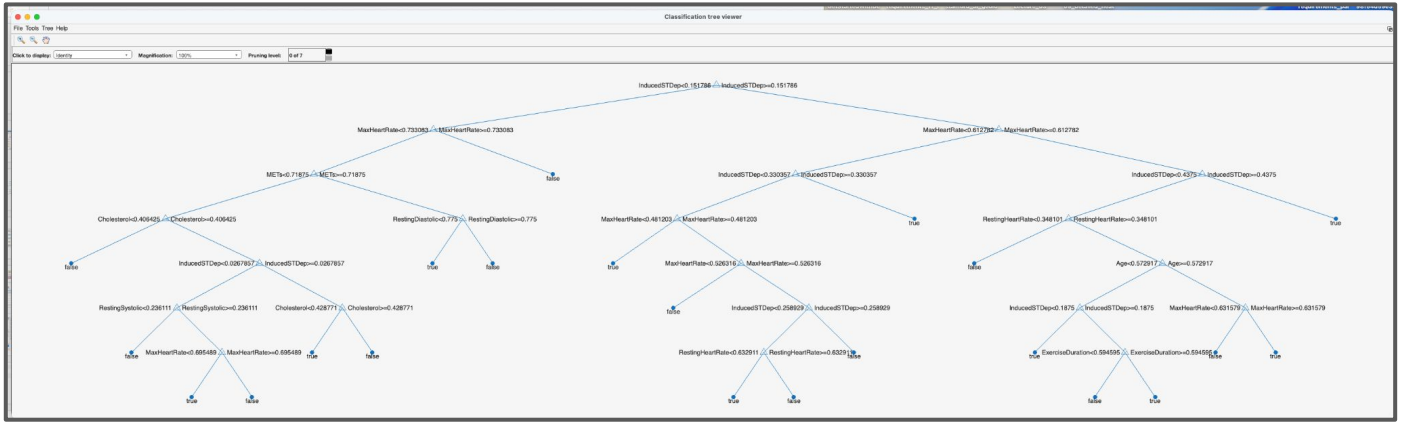
There are several examples provided, e.g.,
Train a generative adversarial network to generate images
https://www.mathworks.com/help/deeplearning/ug/train-generative-adversarial-network.html

If you are interested in using the transformer models, check out this file exchange:
https://mathworks.com/matlabcentral/fileexchange/107375-transformer-models

Livescripts

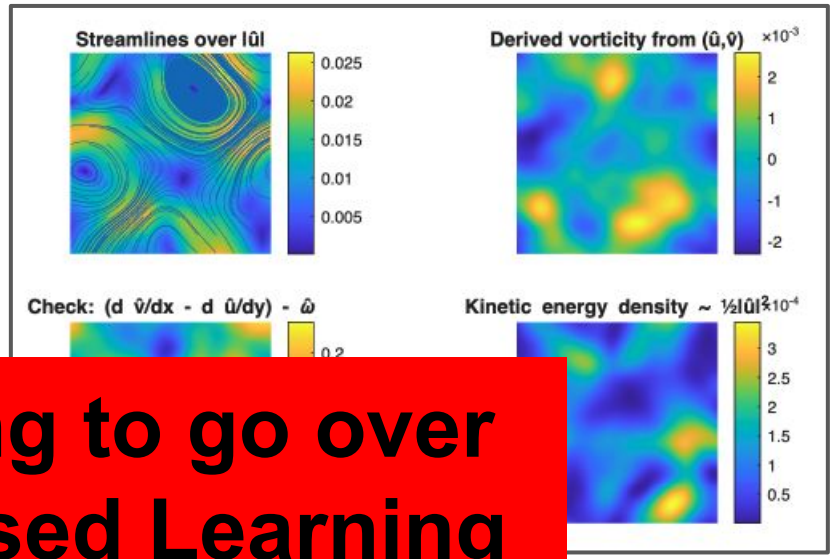
Clustering, Classifiers, Fourier Neural Operators, and more!





Livescripts

Clustering, Clas
Neural Operat



We are going to go over the Supervised Learning Live Script next class due to time constraints.