

Welcome back to **CME 292**
Advanced MATLAB for Scientific Computing
WINTER 2023

Machine Learning with MATLAB

CME 292 LECTURE 4

1/19/2023

Outline

Statistical and Machine Learning

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

Deep Learning

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

Statistical and Machine Learning

Machine Learning with
MATLAB

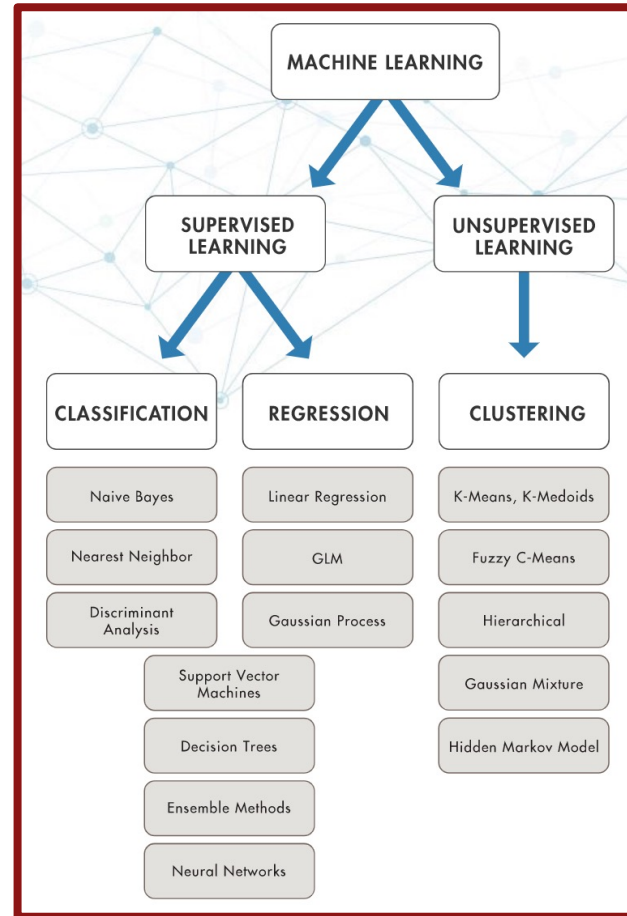


Image from MathWorks

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:

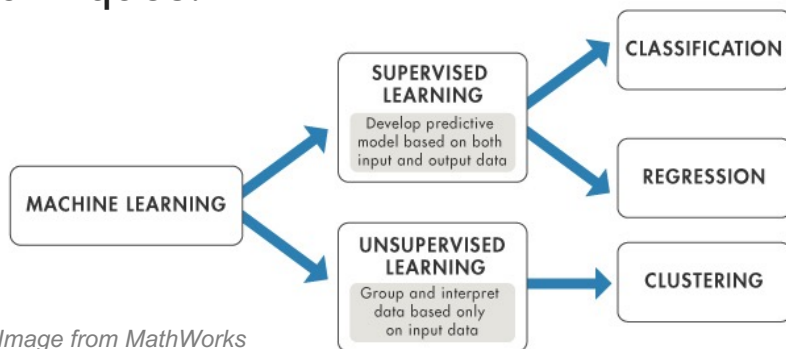


Image from MathWorks

Systematic Machine Learning Workflow

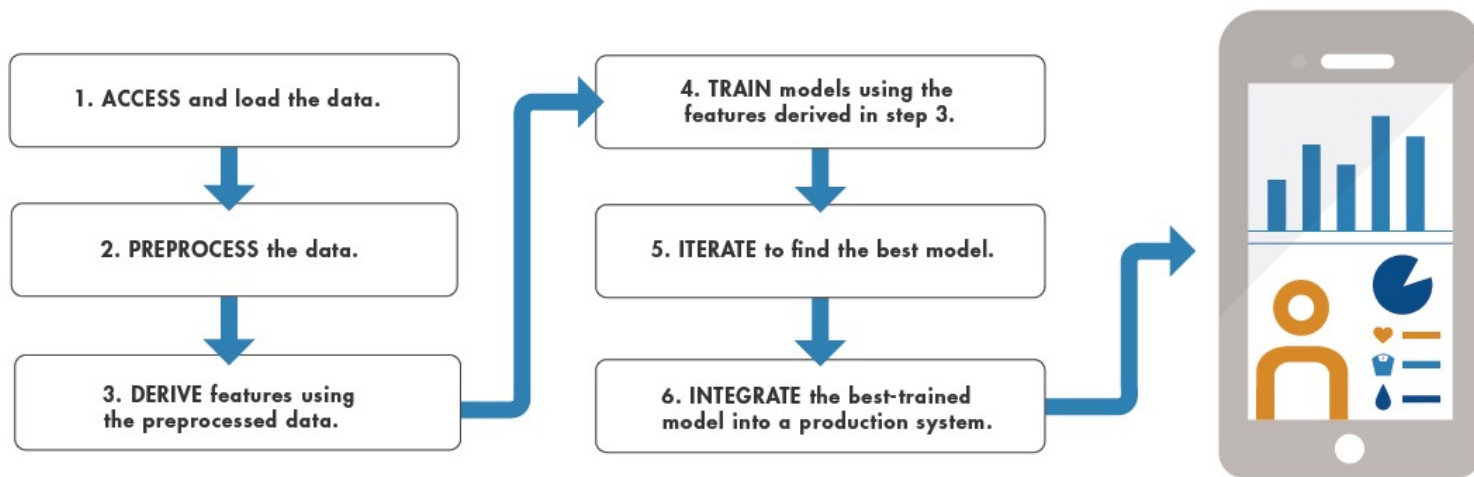


Image from MathWorks

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 data file.
- categorical creates a categorical array from data.
- Assigning the empty array removes rows or columns. rmmmissing 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:

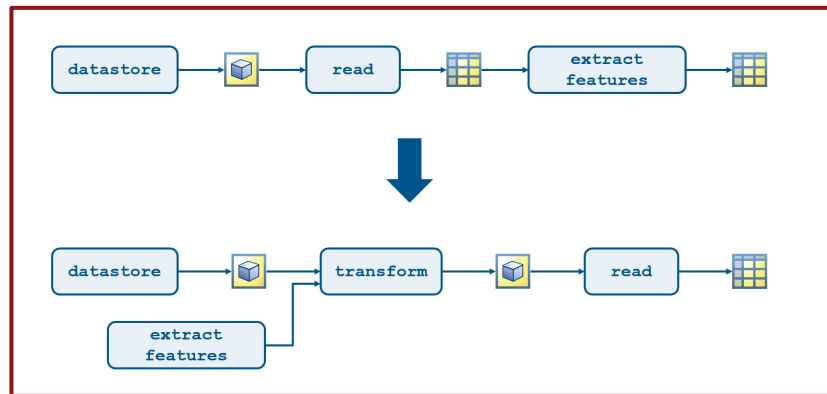
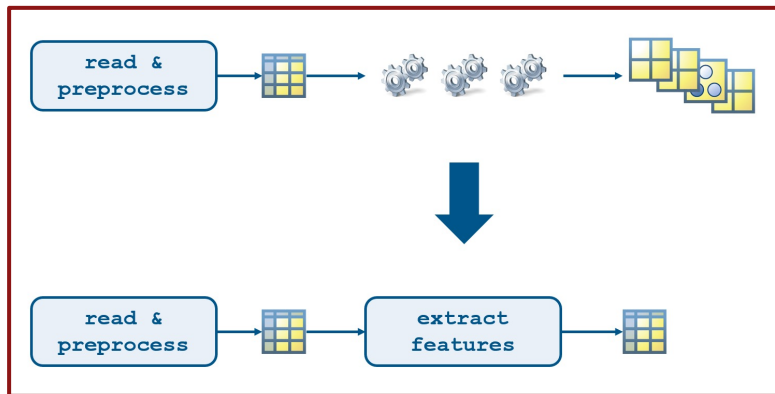
- Finding peaks: `islocalmax`, `islocalmin`
- Correlation between signals: `corr`

Automating Feature Extraction

Custom preprocessing (extraction) functions and transformed datastore

$$ds_{new} = \text{transform}(ds1, ds2, \dots, dsN, @fc_n)$$

where fc_n is the transformation function.



Unsupervised learning

Find hidden patterns or intrinsic structures in data.

Draw inferences from datasets consisting of input data without labeled responses.

- *Clustering* is the most common unsupervised learning technique.
 - find hidden patterns or groupings in data
 - Applications: gene sequence analysis, market research, object recognition, ...
- *Dimensionality reduction* is a key technique in unsupervised learning.

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 q -dimensional space. q is the minimum number of dimensions to achieve the given pairwise distances.
 - `e` Eigenvalues of the matrix $x \times x^T$.

Principal Component Analysis (PCA)

- 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).

Clustering Algorithms

k-means Clustering

```
idx = kmeans(X,k)
```

- *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

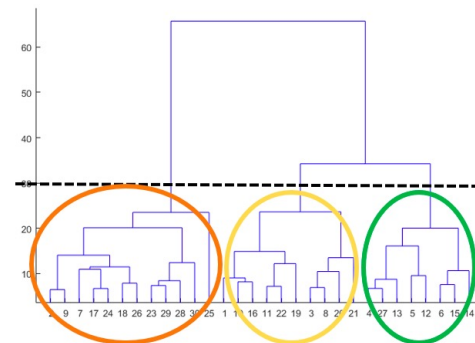
- Fit Gaussian Mixture Model: `gm = fitgmdist(X,2);`
- Identify Clusters: `g = cluster(gm,X); [g,~,p] = cluster(gm,X);`

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 \rightarrow 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

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*

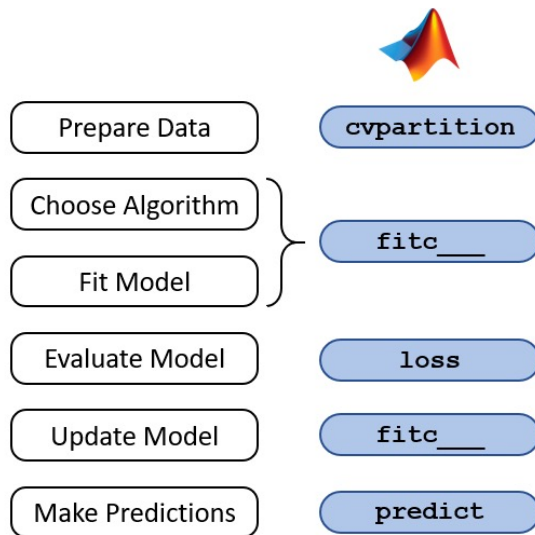
Classification

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

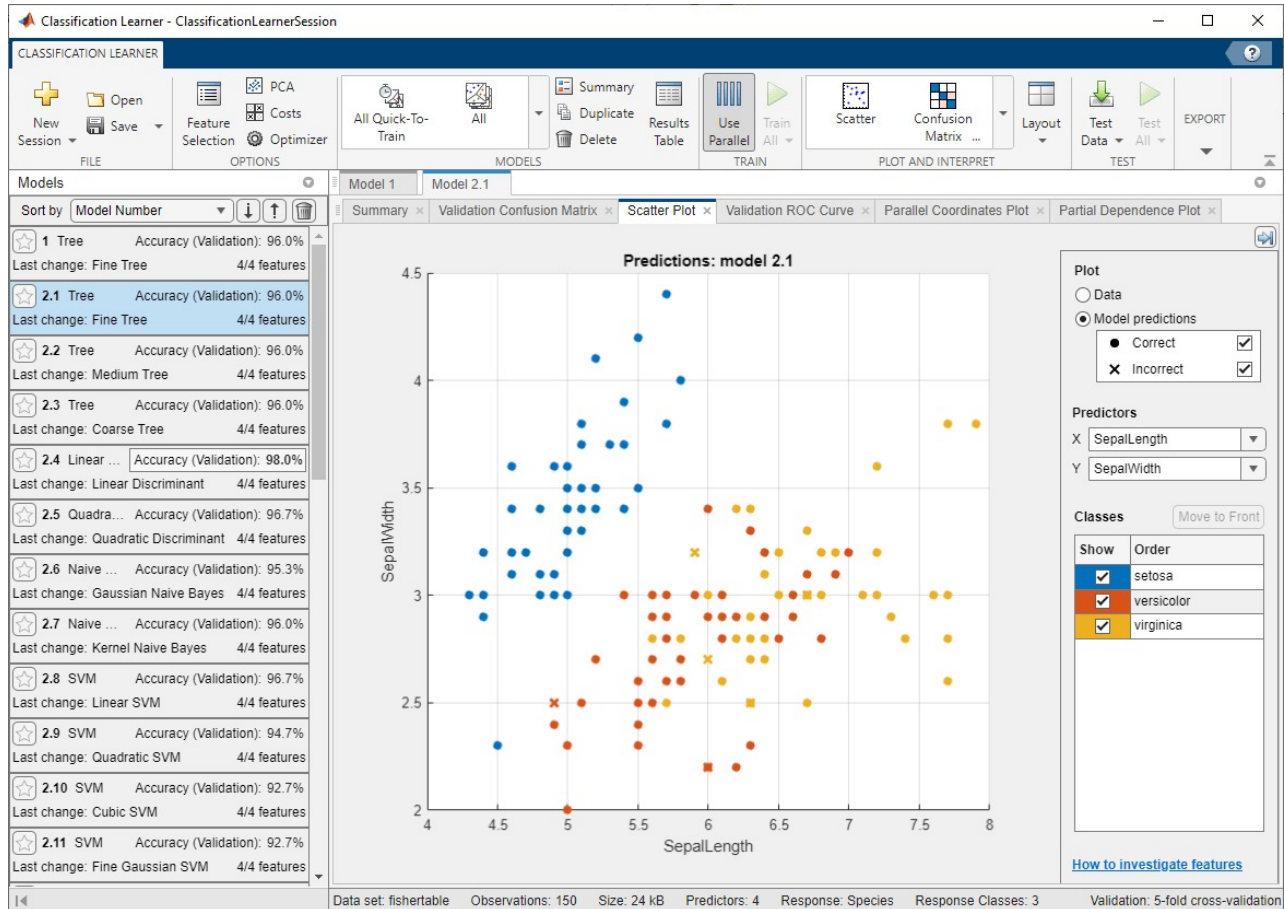
Improving Predictive Models

- Cross Validation
- Hyperparameter Optimization
- Sequential Feature Selection
- Ensemble Learning

Interactive Tool: Classification Learner App

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

- Automatically train a selection of models to help choose the best model.
- 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.



Regression

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

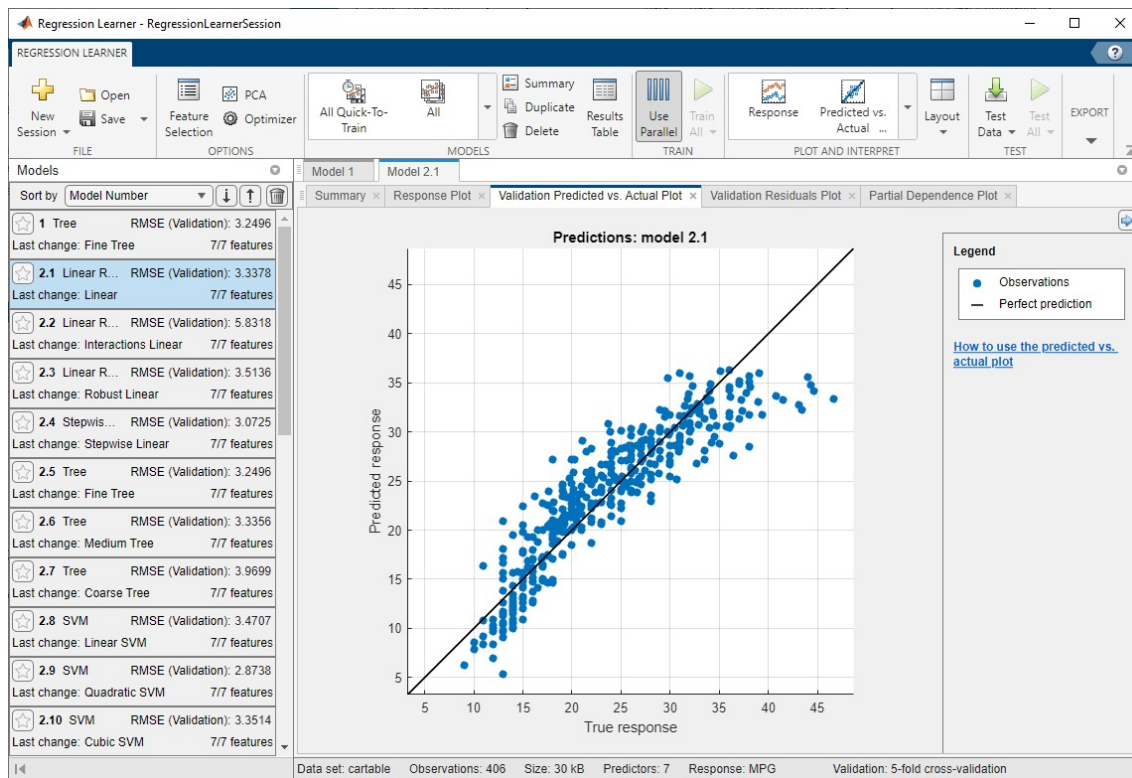
Typical applications:
electricity load forecasting,
algorithmic trading, ...

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*



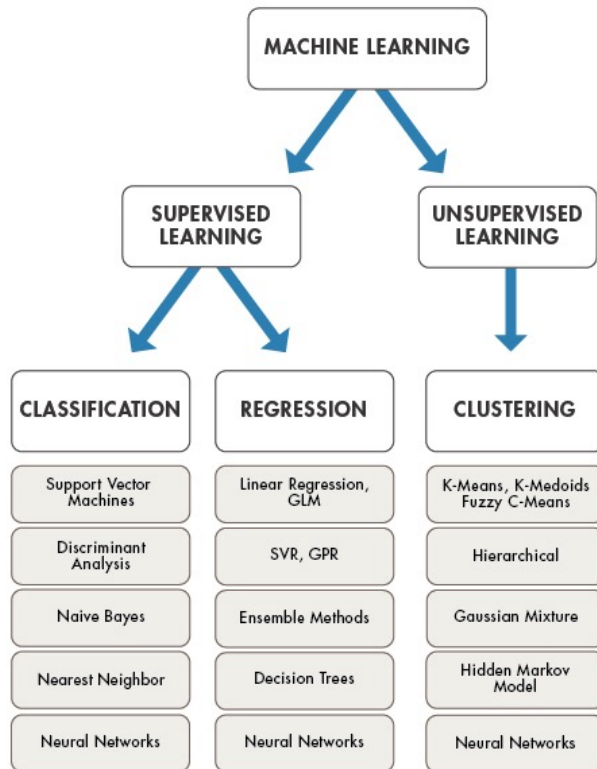
Choose the right algorithm

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



Deep Learning

Machine Learning with
MATLAB

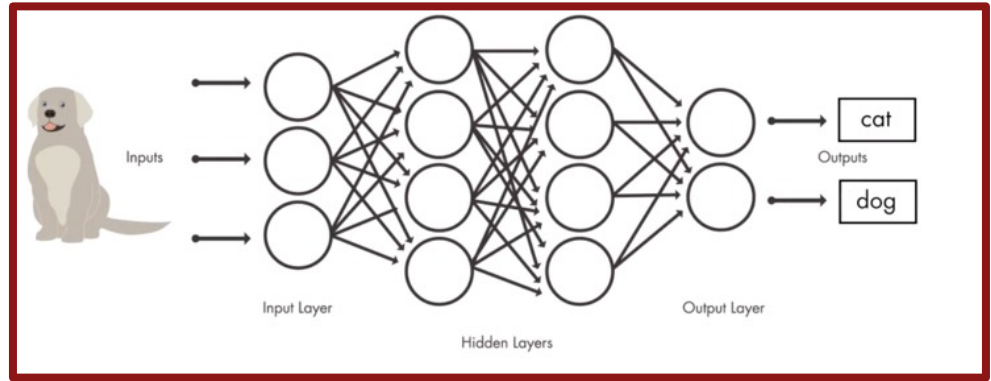
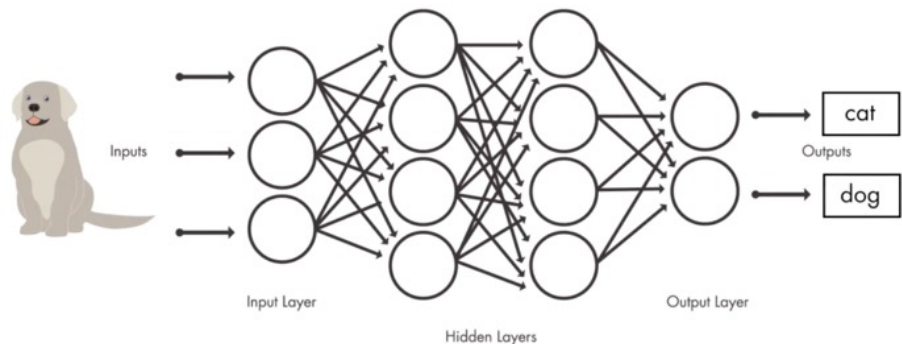


Image from MathWorks

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 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;
```

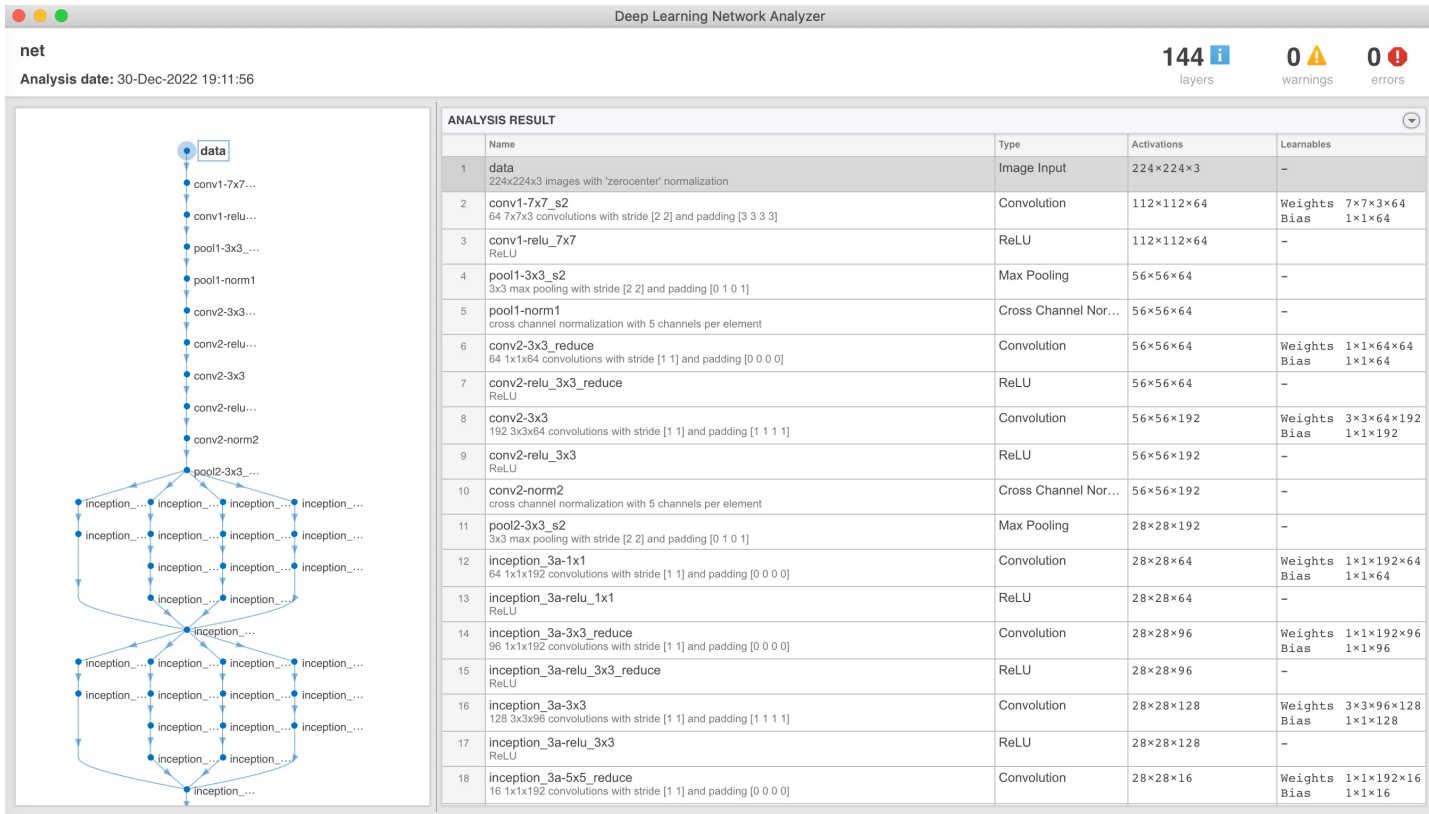
- Display layers of GoogLeNet
`net.Layers`
- View network architecture
`analyzeNetwork(net)`
- Load an image and resize if needed
`I = imread("image.png");`
- Classify the image and display the label.

```
label = classify(net,I);  
figure  
imshow(I)  
title(string(label))
```

bell pepper



Deep Learning Network Analyzer



Deep Learning Visualization Methods

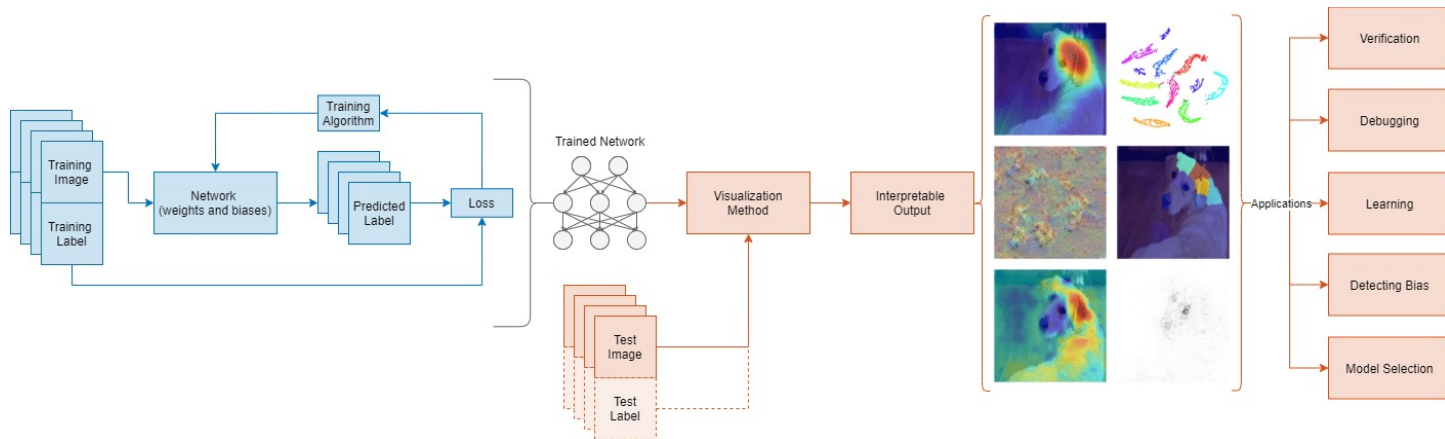
“Black Boxes”: understanding why a network makes a particular decision is crucial.

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)

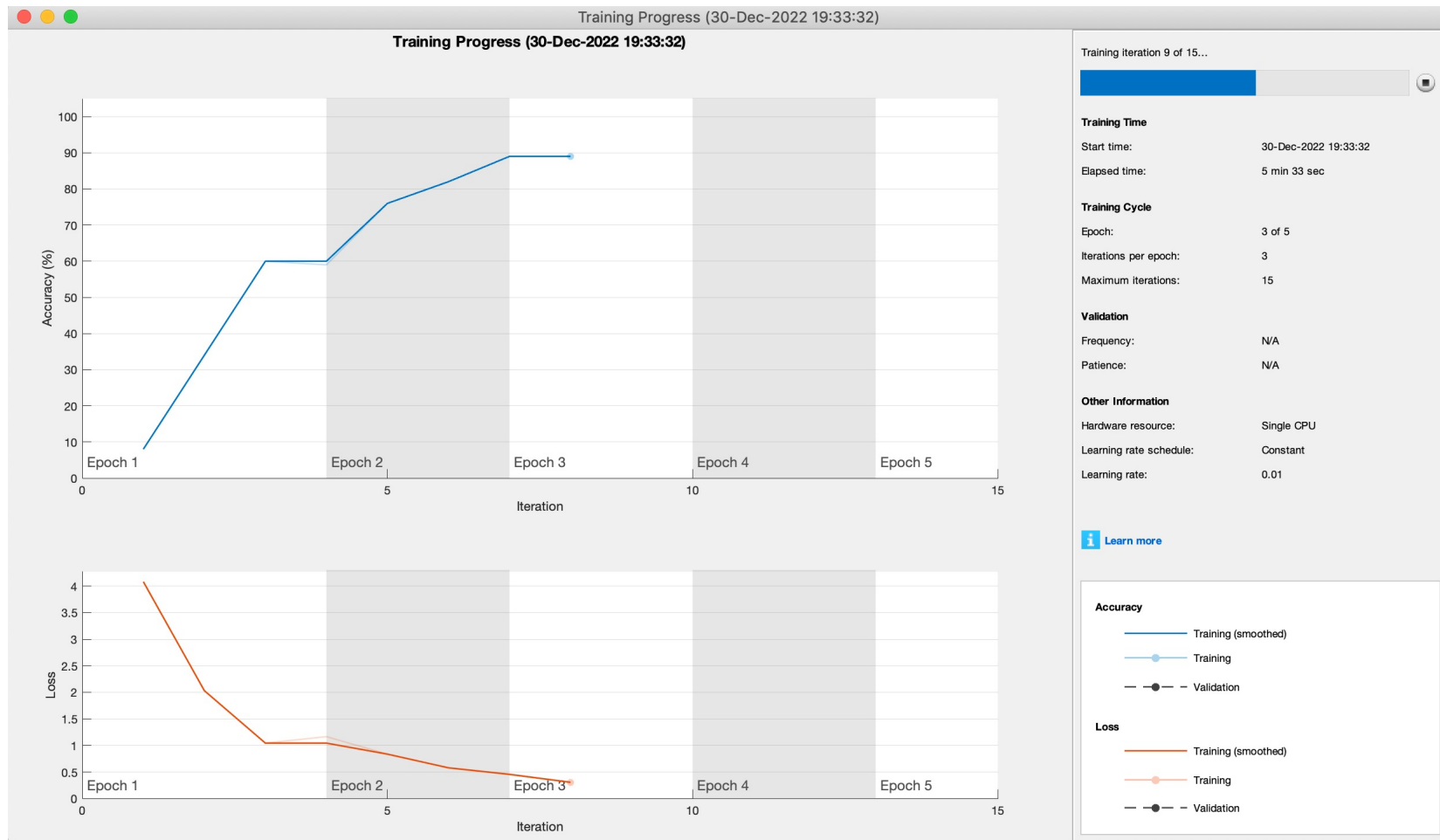
E.g., extract features from an input image using a CNN with the activations function.

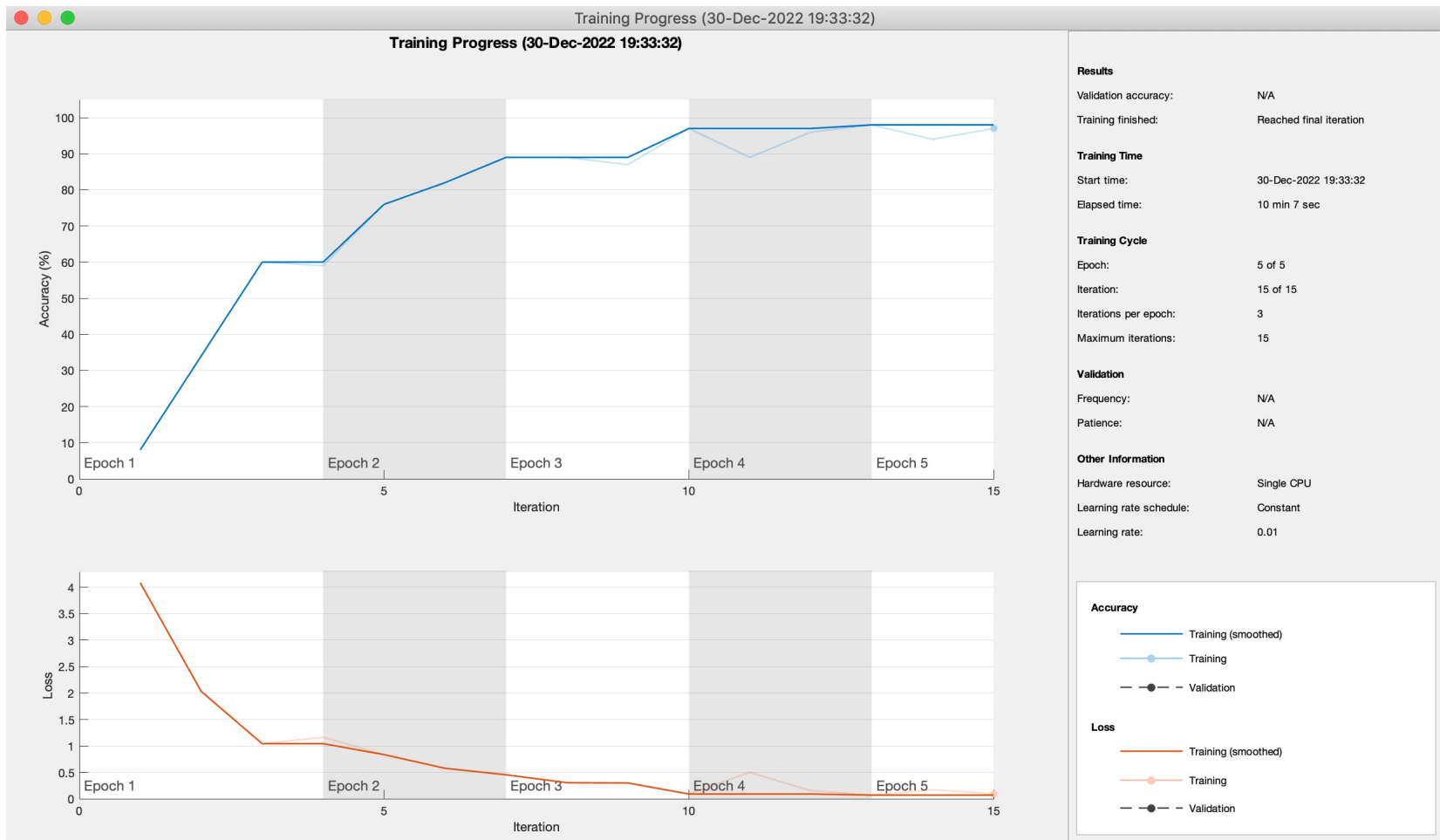
`Y = activations(net, X, layerOut)`

- 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

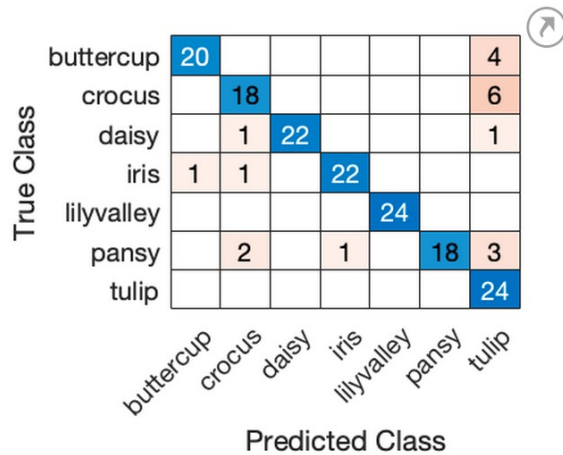
1. Load pretrained model and modify layer(s).
2. Set training algorithm options.
 - An **iteration** is one step taken in the **gradient descent algorithm** towards minimizing the loss function using a mini-batch.
 - An **epoch** is the full pass of the training algorithm over the entire training set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





Accuracy: 0.881

Confusion chart:



A confusion matrix for a flower classification task. The y-axis is labeled 'True Class' and the x-axis is labeled 'Predicted Class'. Both axes list the classes: buttercup, crocus, daisy, iris, lilyvalley, pansy, and tulip. The matrix cells contain counts of instances. Blue cells (diagonal) represent correct classifications: 20 for buttercup, 18 for crocus, 22 for daisy, 22 for iris, 24 for lilyvalley, 18 for pansy, and 24 for tulip. Orange cells represent misclassifications. A small icon with an upward arrow is located to the right of the matrix.

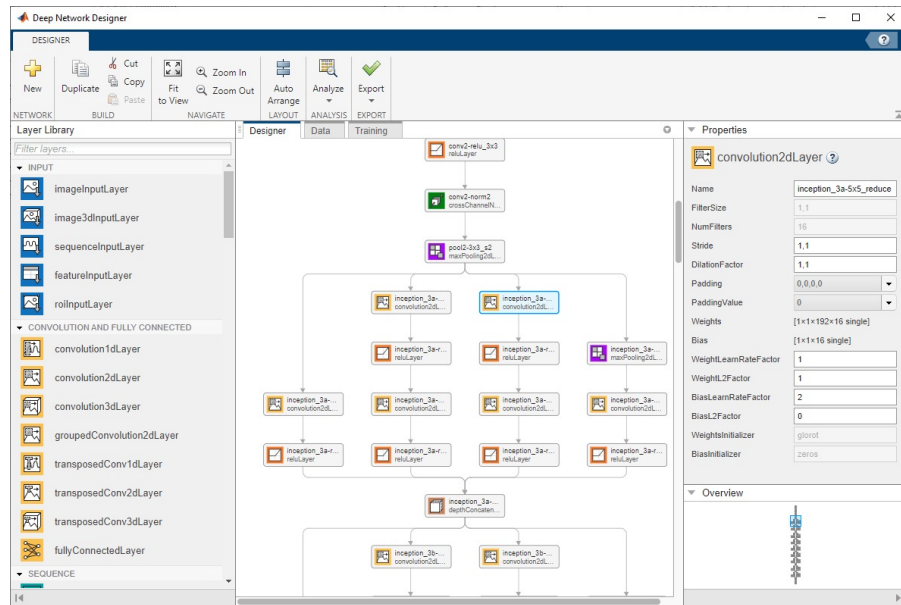
buttercup	20						4
crocus		18					6
daisy		1	22				1
iris	1	1		22			
lilyvalley					24		
pansy		2		1		18	3
tulip							24

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`

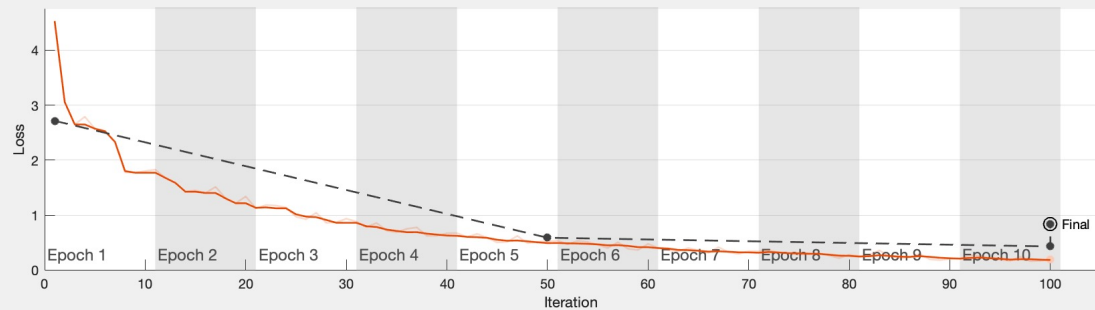
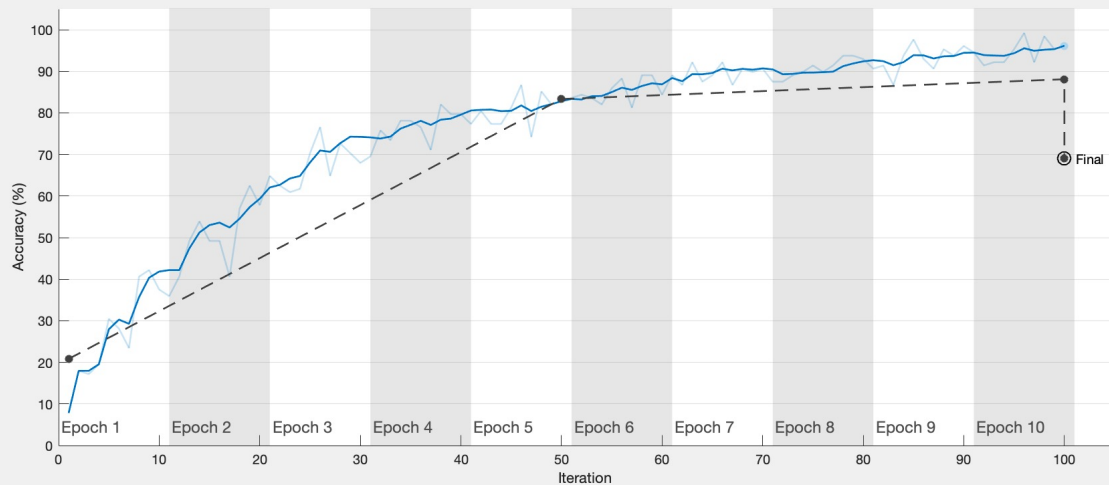
Create Network Architectures from Scratch

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



Training Progress (30-Dec-2022 19:58:37)

Training Progress (30-Dec-2022 19:58:37)



Results

Validation accuracy: 69.05%
Training finished: Reached final iteration

Training Time

Start time: 30-Dec-2022 19:58:37
Elapsed time: 7 min 4 sec

Training Cycle

Epoch: 10 of 10
Iteration: 100 of 100
Iterations per epoch: 10
Maximum iterations: 100

Validation

Frequency: 50 iterations
Patience: Inf

Other Information

Hardware resource: Single CPU
Learning rate schedule: Piecewise
Learning rate: 0.001

Accuracy

— Training (smoothed)
— Training
— Validation

Loss

— Training (smoothed)
— Training
— Validation

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>

Next Lecture

Numerical Linear Algebra

- Sparse matrices
- Matrix decomposition
- Linear system solvers

Numerical Optimization

- Optimization problem
- Nonlinear system of equations
- Optimization solvers

Symbolic Math

ODE and PDE

Fun with MATLAB

Type `fifteen` in the command window.