

Open Source Workshop

Democratizing AI with Open Source and Open Standards



ML TEAM - CODAIT
codait.org

Please go to
ibm.biz/max-workshop

Center for Open Source Data and AI Technologies
(CODAIT)

CODAIT aims to make AI solutions **easier** to
create, deploy, and manage in the enterprise

Relaunch of the Spark Technology Center (STC) to
reflect expanded mission

30+ open source developers!

-

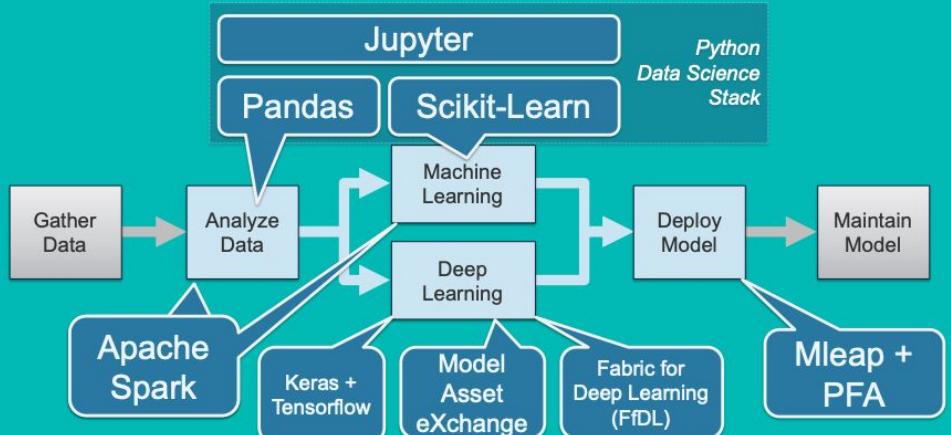
IBM Developer

CODAIT

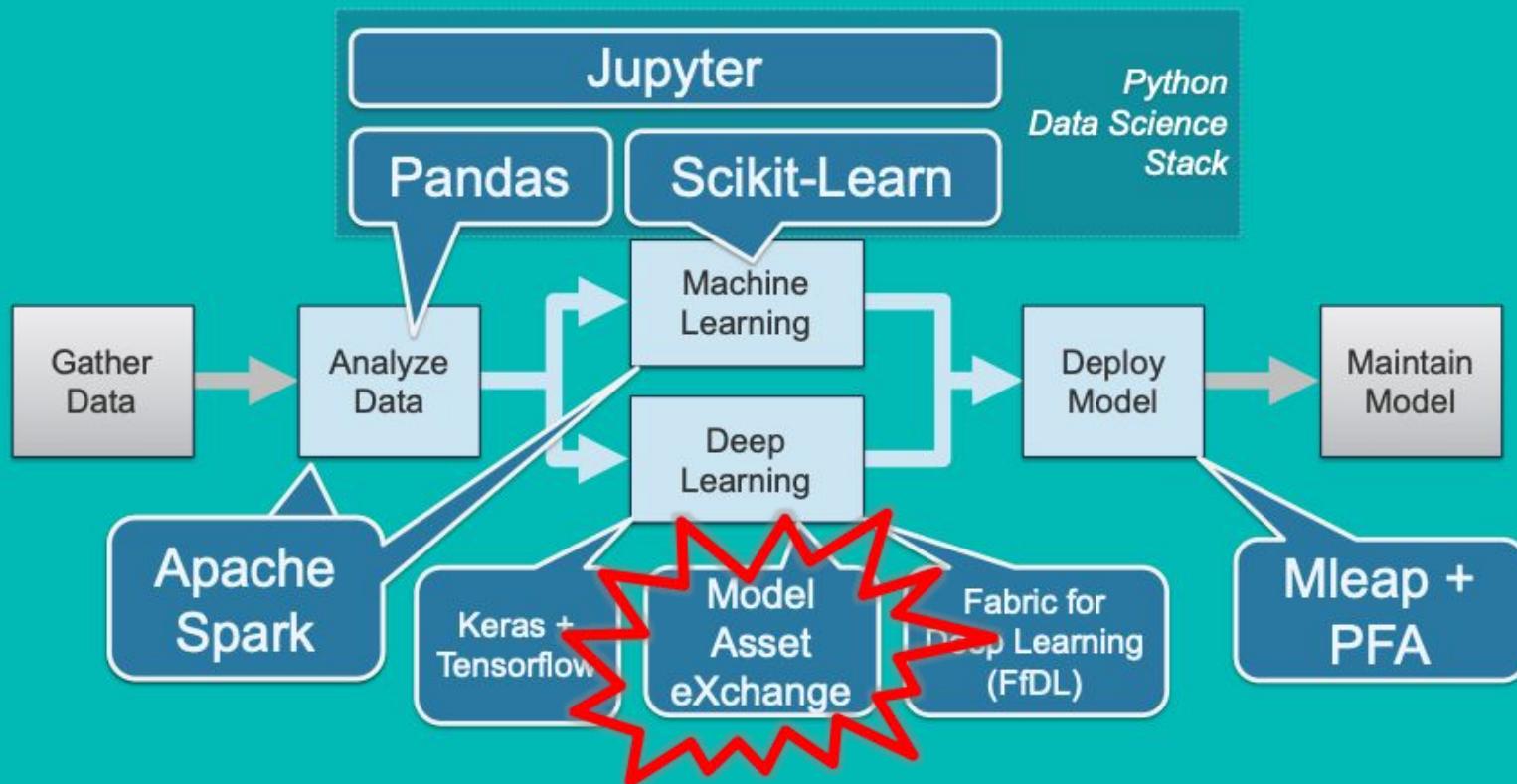
codait.org

Watson West Building
505 Howard St.
San Francisco, California

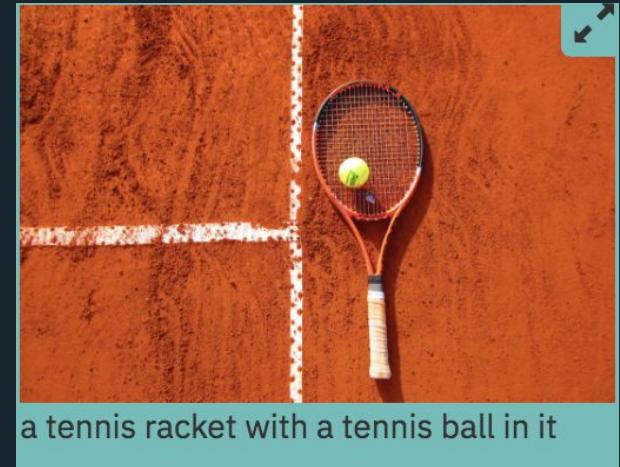
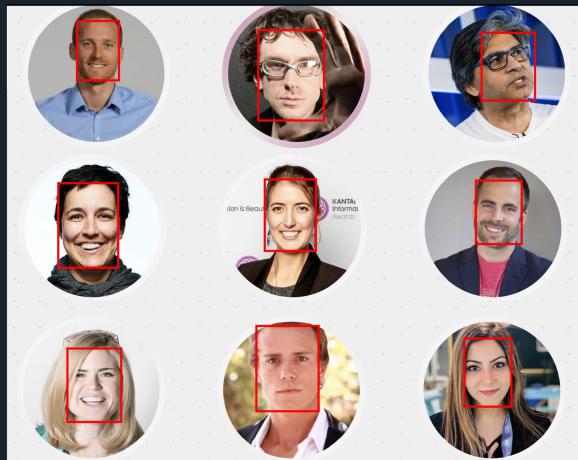
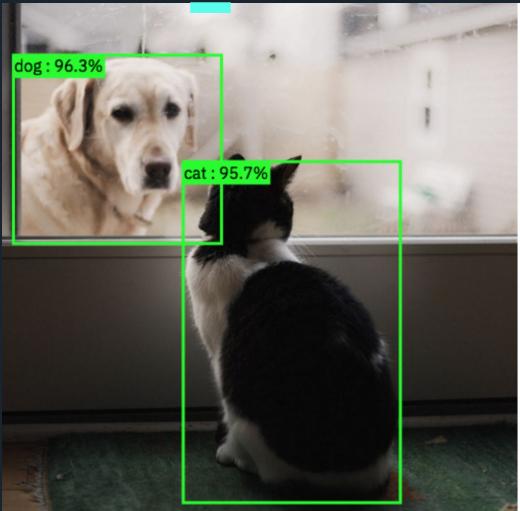
Improving Enterprise AI lifecycle in Open Source



CODAIT: Enabling End-to-End AI in the Enterprise



Have you ever wanted to classify images, recognize faces or places or generate captions of images?



A young child with curly blonde hair, wearing a red t-shirt, is smiling and pulling at the collar of their shirt. The background is a bright, outdoor setting with water and boats.

With the
Model Asset
eXchange,
you can!

The Model Asset eXchange enables **domain experts** to use **deep learning** in the enterprise.

Q: What is deep learning?

A: Machine learning using **deep neural networks**.

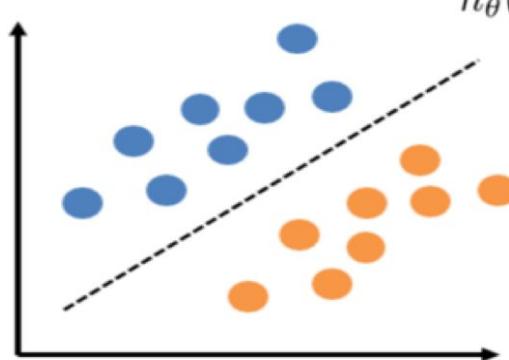
Q: What is a deep neural network?

A: A neural network with multiple hidden layers.

What is a neural network?

Separability: Linear and Non-linear

Linear



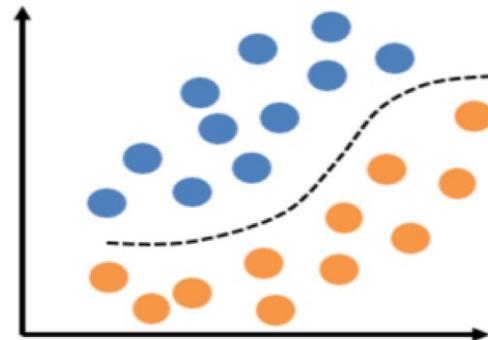
$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

One variable for linearly separable

Hypothesis: $h_{\theta}(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

Nonlinear



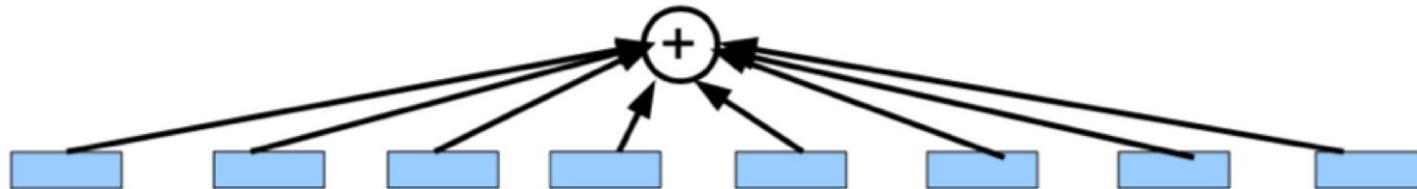
Multiple variables for separability

Hypothesis $h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$

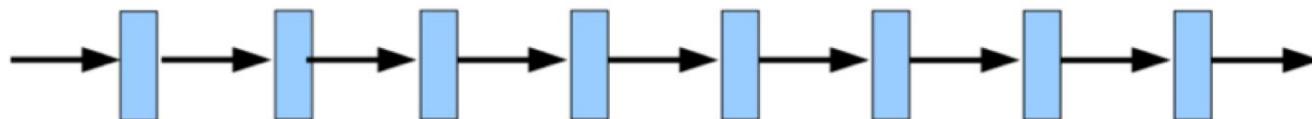
Parameters: $\theta_0, \theta_1, \dots, \theta_n$

Learning Features

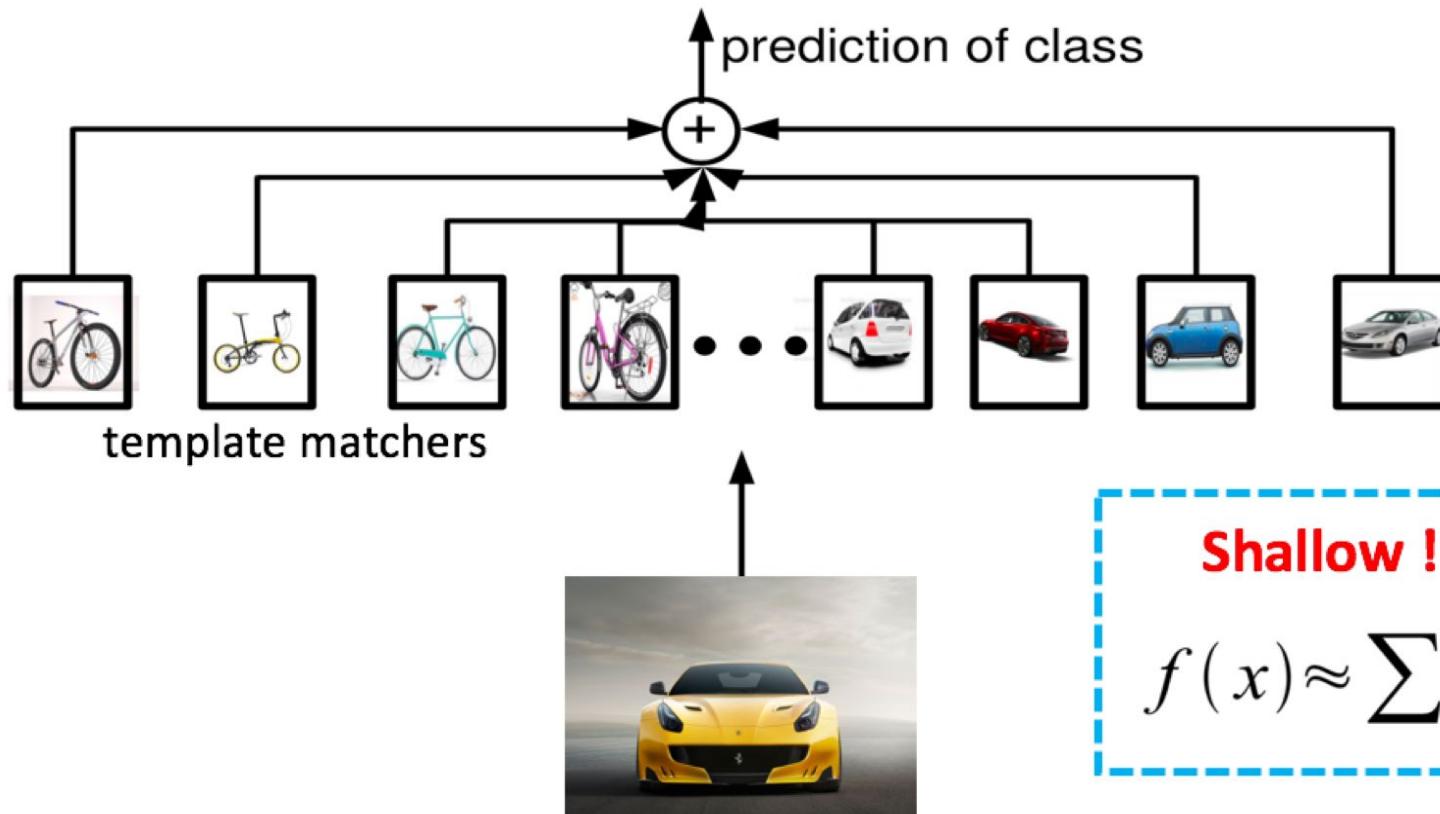
Linear combination : $f(x) \approx \sum_j g_j$



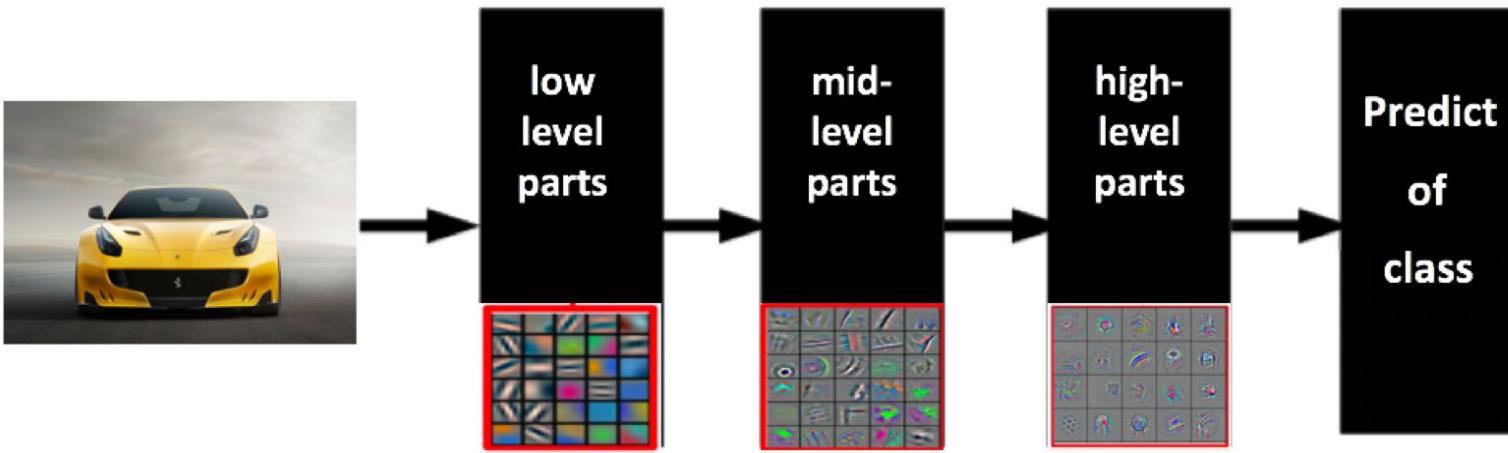
Composition : $f(x) \approx g_1(g_2(\dots g_n(x)\dots))$



Linear Combination



Composition



- **Advantage: intermediate concepts can be re-used**
- Given lots of data => **engineer less** and learn more!

Deep !

$$f(x) \approx g_1(g_2(\dots g_n(x)\dots))$$

Linear Regression

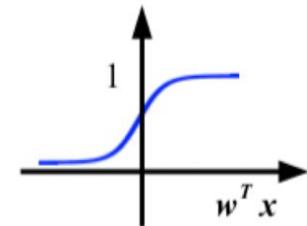
Hypothesis => Logistic function: Generalized Linear Model

Input: $x \in R^D$

Binary label: $y \in [-1, +1]$

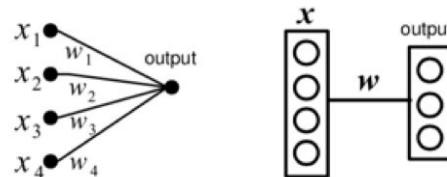
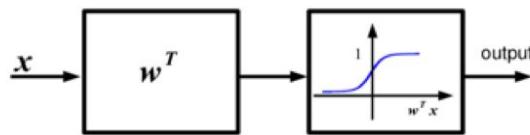
Parameters: $w \in R^D$

Output prediction: $p(y=1|x) = \frac{1}{1+e^{-w^T x}}$



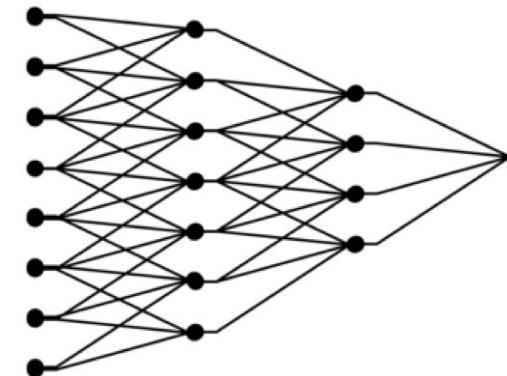
1

Graphical Representation



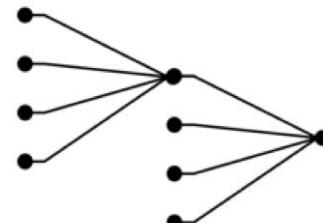
3

Logistic Regression → Neural Nets



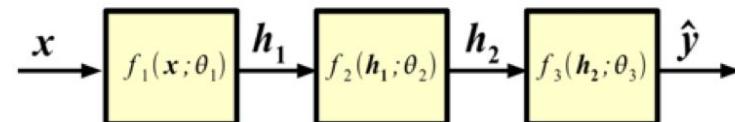
2

Logistic Regression → Neural Nets



4

Neural Nets



NOTE: In practice, each module does **NOT** need to be a logistic regression classifier

What is a neural network?

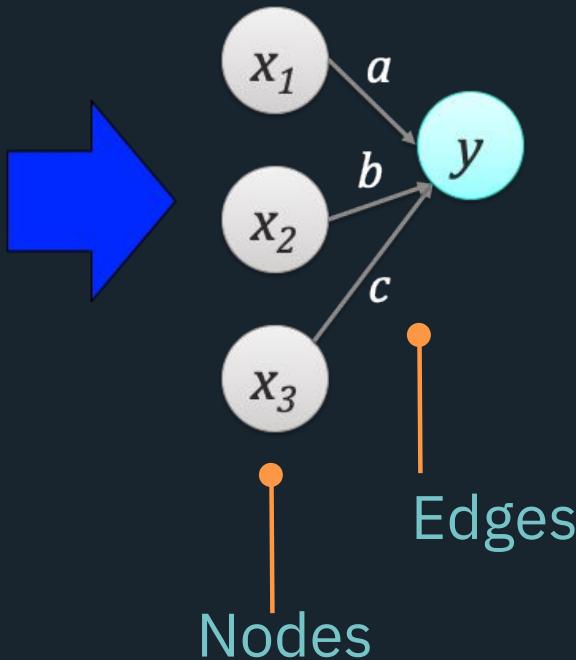
$$y = ax_1 + bx_2 + cx_3$$

Linear regression

What is a neural network?

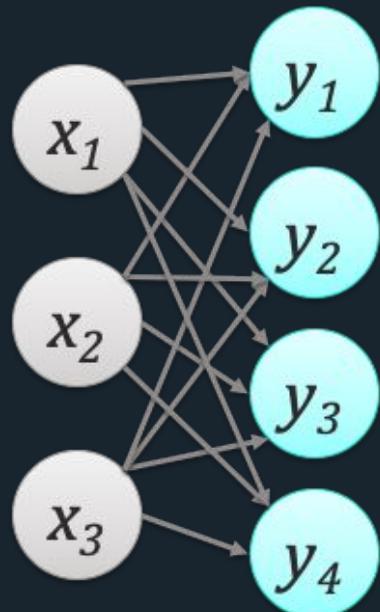
$$y = ax_1 + bx_2 + cx_3$$

Linear regression

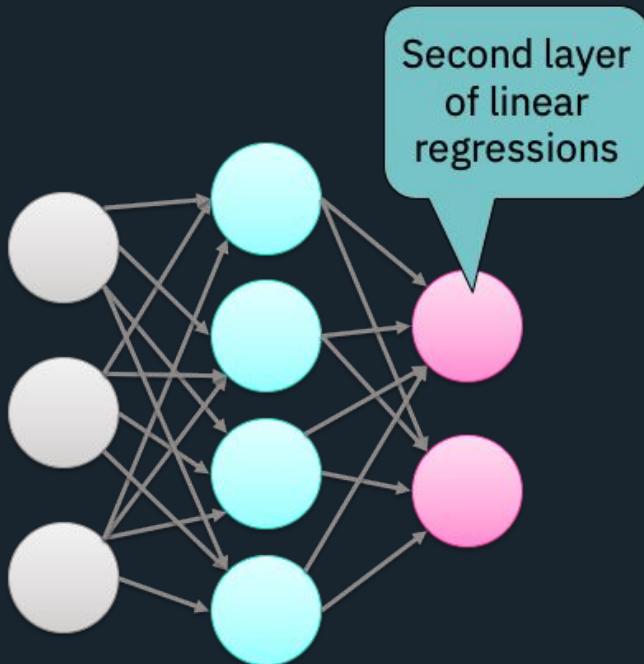


What is a neural network?

Multiple linear regressions at the same time



What is a neural network?

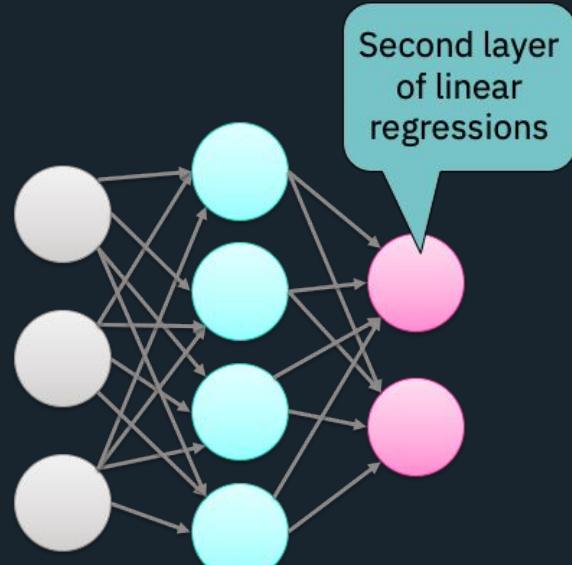


Multilayer Perceptron Neural Network

What is a neural network?



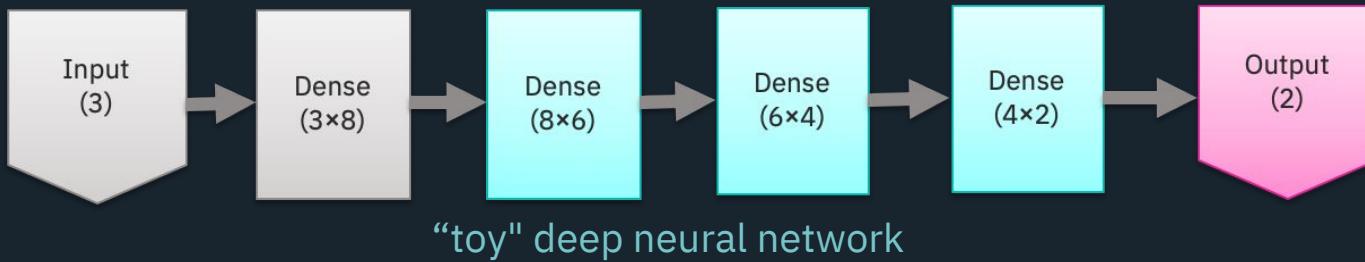
Same network in a more compact notation



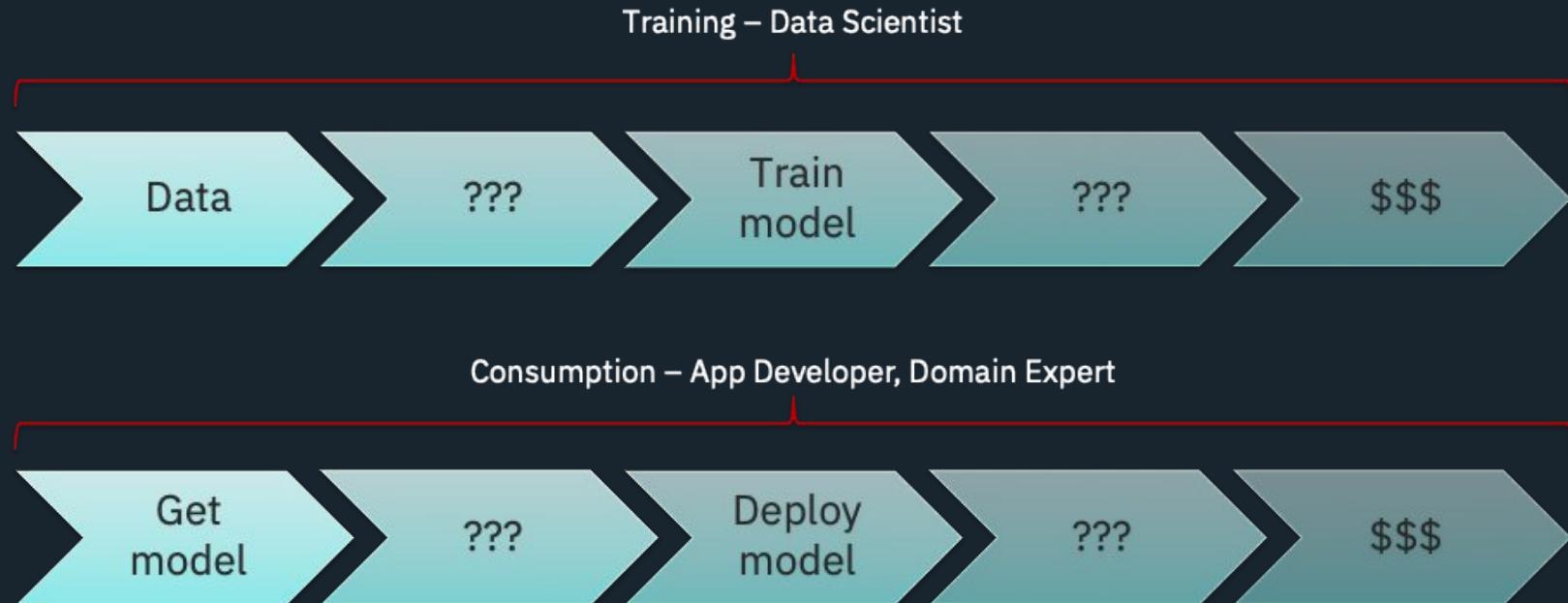
Multilayer Perceptron Neural Network

What is a **deep** neural network?

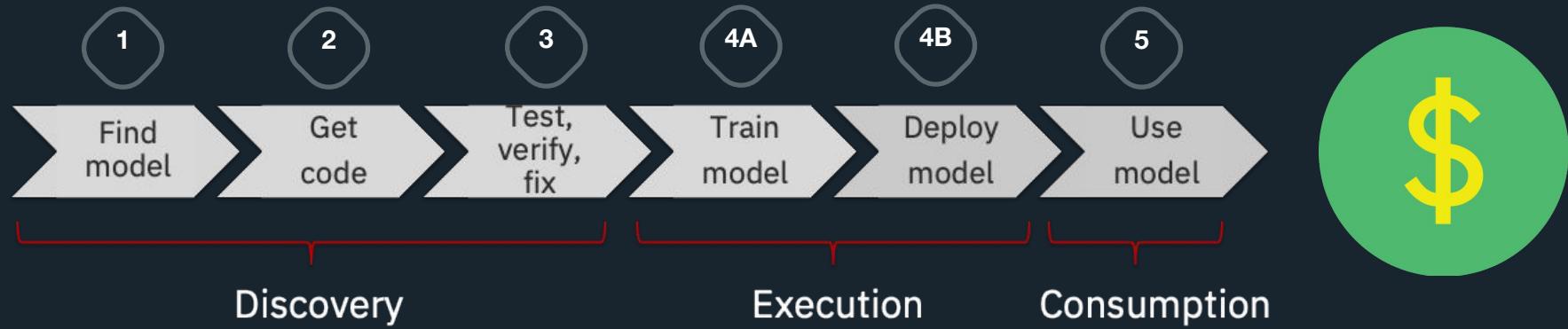
A neural network with **multiple hidden layers**



Applying Deep Learning: Perception



Applying Deep Learning: Reality

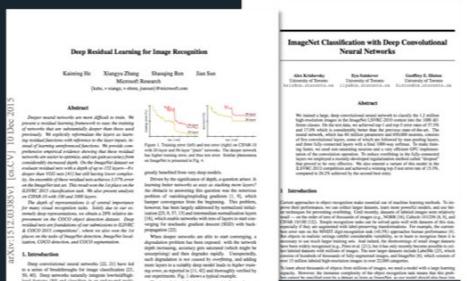
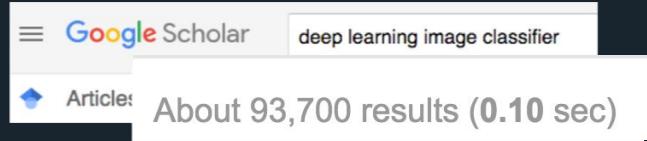


Step 1: Find a model

... that does what you **need**

... that is **free** to use

... that is **performant** enough



Step 2: Get the code

Is there a good **implementation** available?

... that does what you **need**

... that is **free** to use

... that is **performant** enough



TensorFlow code to build ResNet50 neural network graph

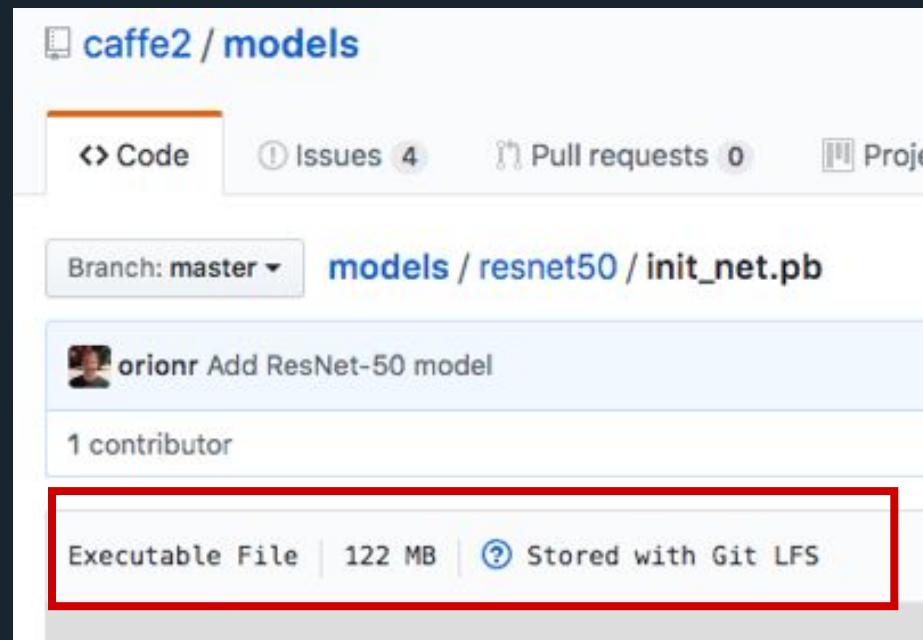
Or ... Step 2: Get the pre-trained weights

Is there a good **pre-trained** model available?

... that does what you **need**

... that is **free** to use

... that is **performant** enough



Step 3: Verify the model you found

Check ...

... that does what you **need**

... that is **free** to use (license)

... that is **performant** enough

(computation & accuracy)



Step 4 (a): Train the model



Step 4 (a): Train the model



Step 4 (b): Figure out how to deploy your model

... adjust **inference** code
(or write from scratch)

... **package** inference code and model code, and pre-trained weights together

... **deploy** your package



Step 5: Consume your model

... **plug in** into your application

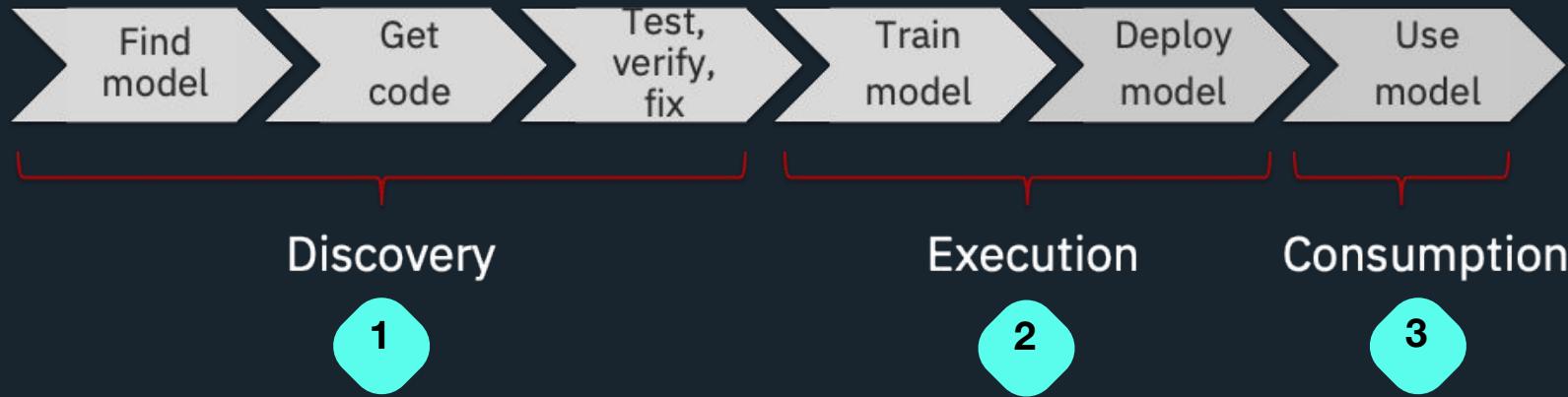


Step 6: Profit

... hopefully



Applying Deep Learning: Reality



Model Asset eXchange

- A one-stop place for developers/data scientists to find and use **free** and **open source** deep learning models

ibm.biz/model-exchange

The screenshot shows the homepage of the Model Asset eXchange. At the top, there's a banner with a man wearing glasses and a beard. Below the banner, the text "Free, deployable, and trainable code." and "A place for developers to find and use free and open source deep learning models." is displayed. There are two buttons: "View all models >" and "Try the tutorial >". Below these buttons, there are three tabs: "Featured", "Deployable" (which is selected), and "Trainable". The main content area displays six model cards arranged in a grid:

Category	Model Name	Description	View Model	Tags
Deployable Facial Recognition	Facial Emotion Classifier	Detect faces in an image and predict the emotional state of each person.	View model »	(Artificial intelligence) (Docker) +
Deployable Image-to-Image Translation Or Transformation	Image Completer	Recognize and extract faces in an image and complete the corrupted portions.	View model »	(Artificial intelligence) (Docker) +
Deployable Object Detection In Images	Human Pose Estimator	Detect humans in an image and estimate the pose for each person.	View model »	(Artificial intelligence) (Docker) +
Deployable Named Entity Recognition	Named Entity Tagger	Locate and tag named entities in text.	View model »	(Artificial intelligence) (Docker) +
Deployable Facial Recognition	Facial Recognizer	Recognize faces in an image and extract embedding vectors for each face.	View model »	(Artificial intelligence) (Docker) +
Deployable Facial Recognition	Facial Age Estimator	Recognize faces in an image and estimate the age of each face.	View model »	(Artificial intelligence) (Docker) +

Model Asset eXchange

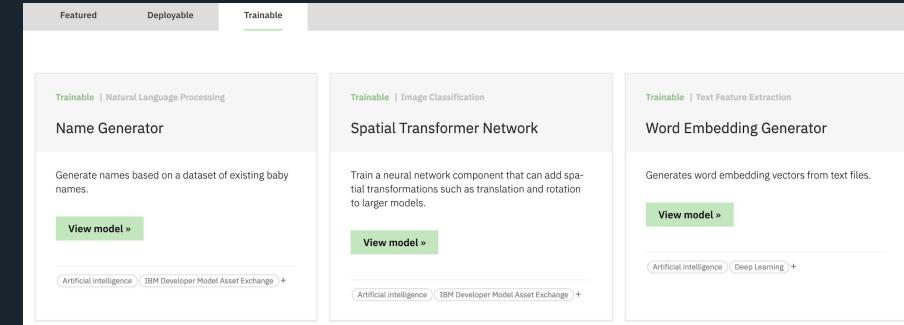
- Wide variety of domains (text, audio, image, etc)
- Multiple deep learning frameworks
- Vetted and tested code and IP
- Build and deploy a model web service in seconds
- Training on [Fabric for Deep Learning \(FfDL\)](#) or [Watson Machine Learning](#) in minutes

ibm.biz/model-exchange

The screenshot shows the homepage of the Model Asset eXchange. At the top, there's a banner with a man wearing glasses and a beard. Below the banner, the text reads "Free, deployable, and trainable code. A place for developers to find and use free and open source deep learning models." There are two buttons: "View all models" and "Try the tutorial". Below the banner, there are three tabs: "Featured", "Deployable" (which is selected), and "Trainable". The main content area displays six model cards, each with a "Deployable" badge and a brief description. Each card has a "View model" button and a "Artificial intelligence | Docker" badge.

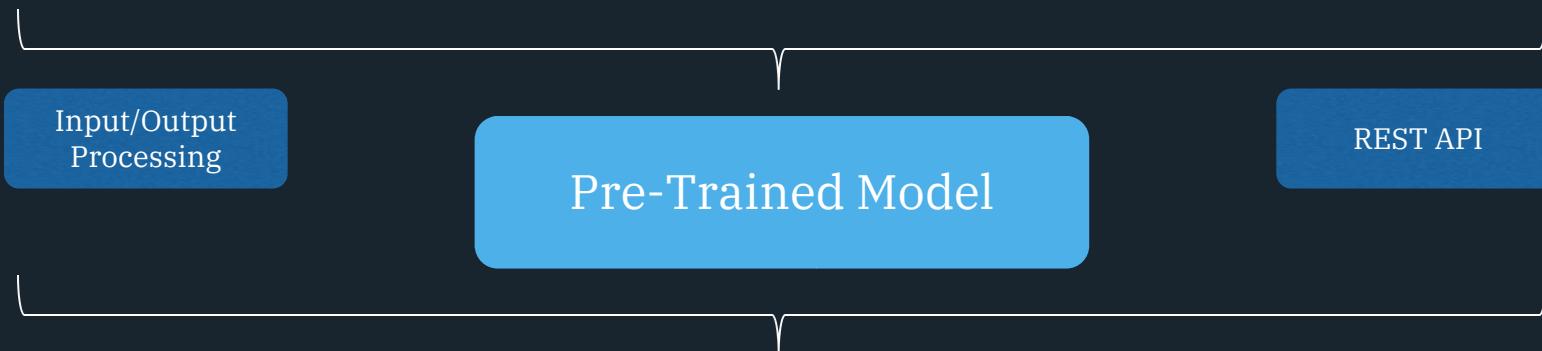
Model Type	Description	Category
Deployable	Facial Recognition	Facial Emotion Classifier
Deployable	Image-to-Image Translation Or Transformation	Image Completer
Deployable	Object Detection In Images	Human Pose Estimator
Deployable	Named Entity Recognition	Named Entity Tagger
Deployable	Facial Recognition	Facial Recognizer
Deployable	Facial Recognition	Facial Age Estimator

Trainable Models



<https://github.com/IBM/FfDL>

Deployable Models



Deep Learning Asset on Model Asset Exchange
ibm.biz/model-exchange

Deployable Models

Deep Learning Asset on Model Asset Exchange
ibm.biz/model-exchange

Deploy

Microservice

Swagger
Specification

Inference
Endpoint

Metadata
Endpoint

REST API

OPEN SOURCE
&
FREE

ibm.biz/model-exchange



Highlights

- Image, audio, text, healthcare, time-series and more
- Pre- / post-processing & inference wrapped up in Docker container
- Generic API framework code - Flask RESTPlus
- Swagger specification for API
- One-line deployment locally and on a Kubernetes cluster

MAX Object Detector 1.1.0

[Base URL: /]
<http://0.0.0.0:5000/swagger.json>

Localize and identify multiple objects in a single image.

model Model information and inference operations

GET /model/labels Return the list of labels that can be predicted by the model

GET /model/metadata Return the metadata associated with the model

POST /model/predict Make a prediction given input data

Parameters

Cancel

Name	Description
image * required	An image file (encoded as PNG or JPG/JPEG)
file (formData)	<input type="file"/> Choose File No file chosen
threshold number (query)	Probability threshold for including a detected object in the response in the range [0, 1] (default: 0.7). Lowering the threshold includes objects the model is less certain about. <input type="text"/> 0.7

Execute

Responses

Response content type application/json

Code	Description
200	Success

Example Value Model

Code Patterns demonstrating how to easily consume MAX models

IBM Developer Topics ▾ Community ▾ More open source at IBM ▾ Search 

Code Patterns

Technology ▾ Industry ▾ Deployment Models ▾ Sort by: Newest First ▾

[IBM Developer Model Asset Exchange !\[\]\(0ae8361765f665634e41f95a70e8b74b_img.jpg\)](#)

CODE PATTERN | SEP 20, 2018

Create a web app to show the age estimation from the detected human faces

[Get the Code »](#)

Artificial Intelligence Docker +

CODE PATTERN | SEP 05, 2018

Create a web app to visually interact with objects detected using machine learning

[Get the Code »](#)

Artificial Intelligence Deep Learning +

CODE PATTERN | JUL 13, 2018

Train and evaluate an audio classifier

[Get the Code »](#)

Artificial Intelligence Deep Learning +

CODE PATTERN | JUL 12, 2018

Create a web app to interact with machine learning generated image captions

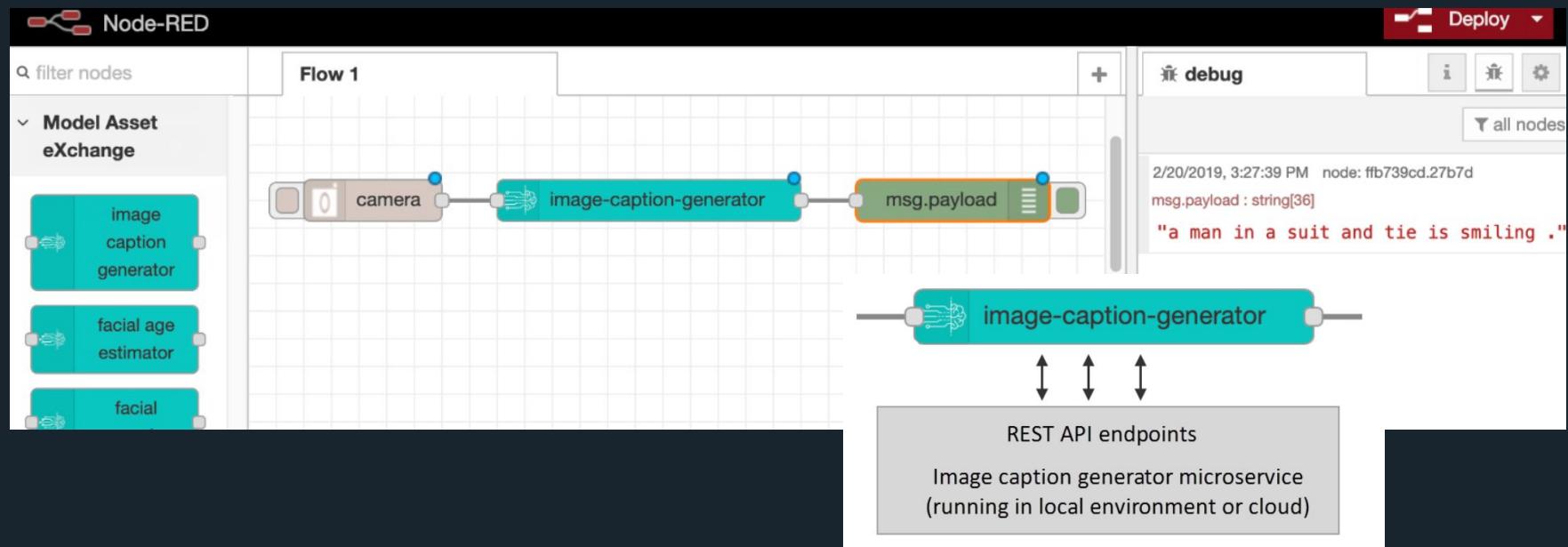
[Get the Code »](#)

Artificial Intelligence Docker +

ibm.biz/max-developers

Leveraging MAX models in IOT scenarios

- Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.
- The [node-red-contrib-model-asset-exchange](#) module adds support for MAX deep learning models.

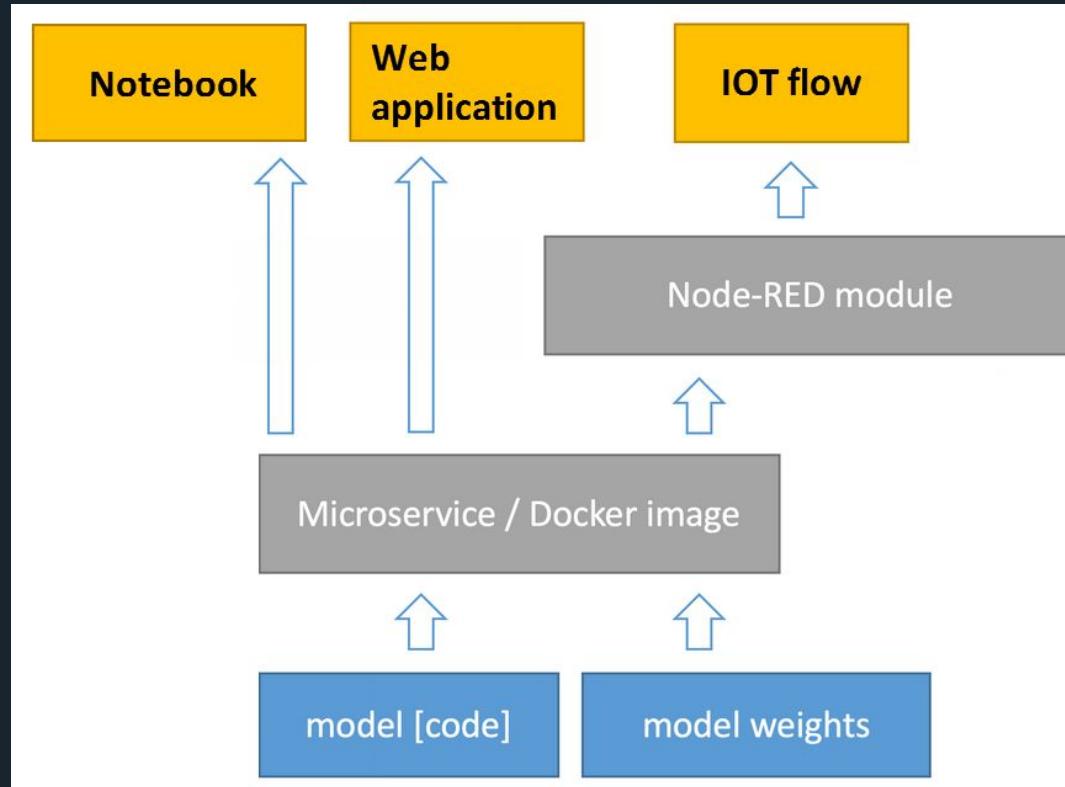


Leveraging MAX models in IOT scenarios

- MAX nodes communicate with the corresponding microservice
- Learn more: <http://ibm.biz/max-for-node-red>



Example consumption scenarios you can try today



Part 2: Hands-On

Getting Started with MAX



Which Deep Learning models will we cover?

Object Detection

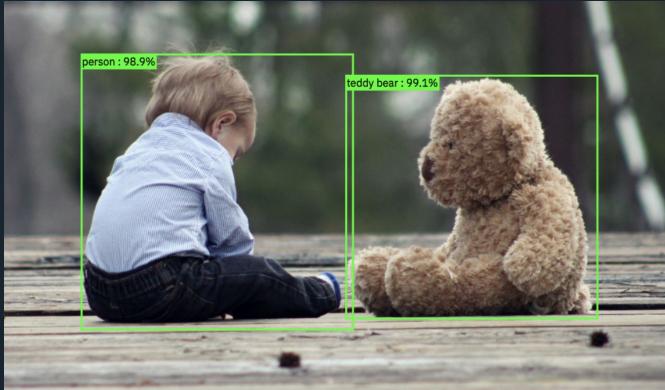


Image Style Transfer



Caption Generation



A man riding a wave on a surfboard

Pose Estimation



And many more!

Schedule

1. Installing Docker and spinning up the MAX Object Detector (15 min)

2. Using the Object Detector WebApp (15 min)

-- BREAK (10 min) --

3. Integrating a Deep Learning model in your own WebApp

- Exploring other MAX models: querying the APIs (30 min)
- Creating your own DL WebApp (30 min)

4. (Optional) MAX models for IOT using Node-RED

Please go to **ibm.biz/max-workshop**

WebApp Creation Flow

1. Clone the MAX tutorial repository

```
git clone https://github.com/IBM/max-tutorial-app-python.git
```

2. Open app.py in an editor or IDE

3. Complete `TODO T1` and `TODO T2`

```
# TODO T1: replace model URL placeholder  
model_url = args.ml_endpoint.rstrip('/') + '**TODO**'
```

Solution: `/model/predict`

```
# TODO T2: uncomment next line and replace placeholder  
# result = output_data['**TODO**']
```

Solution: `predictions`



