

Persian MNIST in 5 Minutes

Shahriar Yazdipour

Dr. Borna

Agenda

Quick Intro to ML

What is MNIST

What is Tensorflow?

Building Model with Python

MNIST on Android.Kotlin

Building MNIST Model

Models Differences

ML On All Platform



Quick Intro to ML

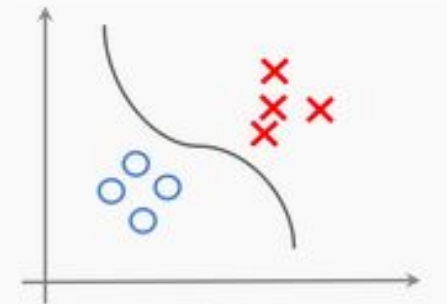
Quick Intro to ML

- Machine learning: A facet of AI that focuses on algorithms, allowing machines to learn without being programmed and change when exposed to new data.
- Deep learning: The ability for machines to autonomously mimic human thought patterns through artificial neural networks composed of cascading layers of information.

Quick Intro to ML

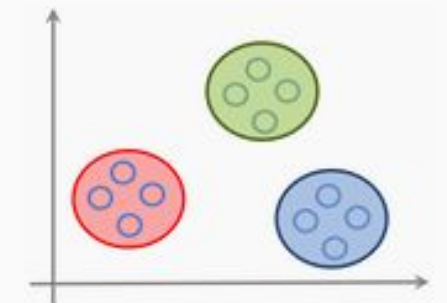
- Supervised Learning:

- **Prediction based on Labelled Data**
- Goal: A program that performs a task as good as humans.
- TASK: well defined
- EXPERIENCE: training data provided
- PERFORMANCE: error/accuracy on the task



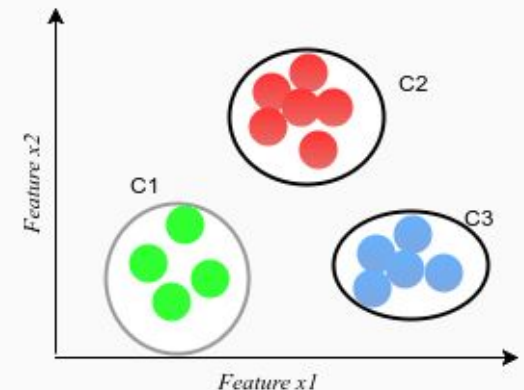
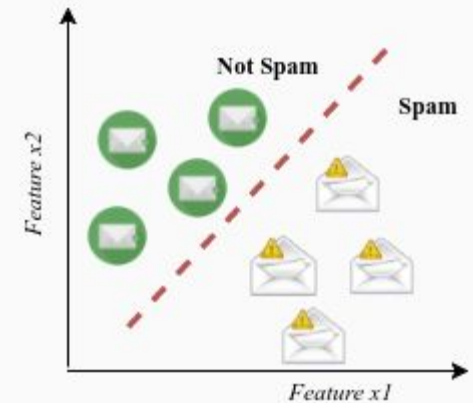
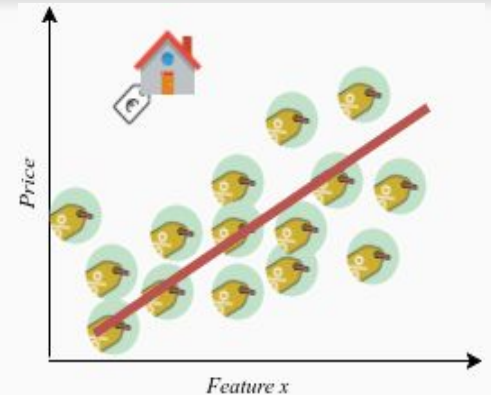
- Unsupervised Learning:

- **Analyze Unlabelled Data**
- GOAL: To find structure in the data
- TASK: vaguely defined (approximate)
- No EXPERIENCE
- No PERFORMANCE



Quick Intro to ML

- Regression (Supervised Learning):
a set of statistical processes for estimating the relationships among variables.
(Linear/Polynomial Regression)
- Classification (Supervised Learning):
Algorithms that let machines assign a category to a data point based on training data.
(KNN, Trees, ...)
- Clustering (Unsupervised Learning):
Algorithms that let machines group data points or items into groups with similar characteristics.
(KMeans, SVD, PCA) + Dimensionality Reduction

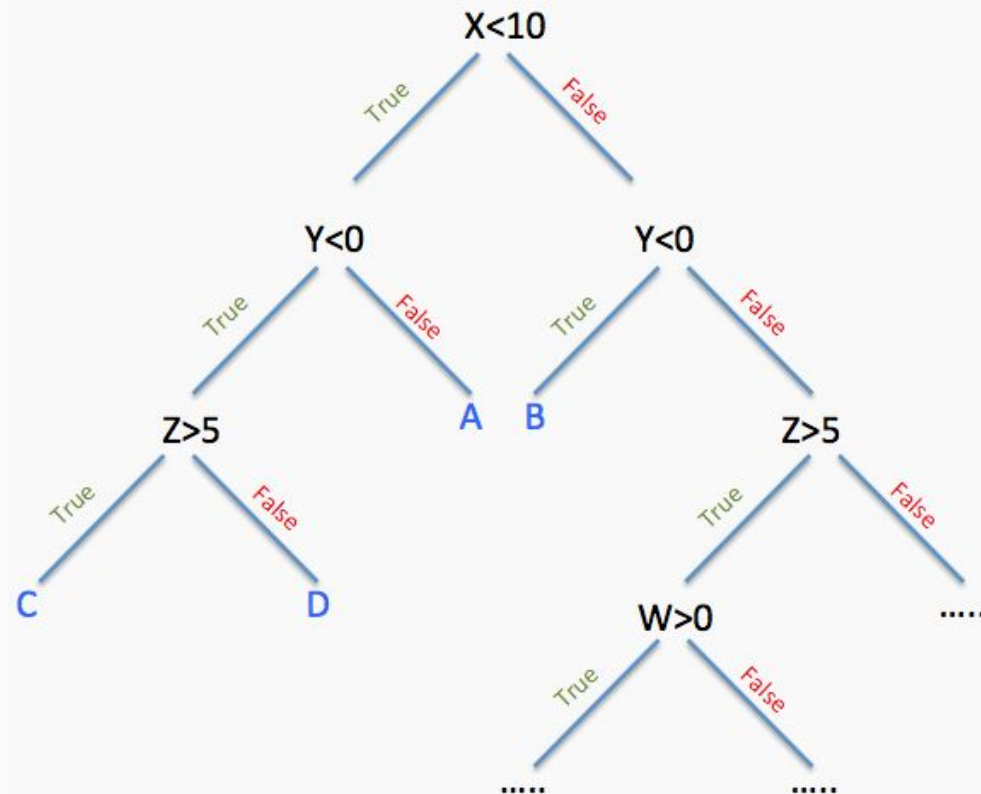


Quick Intro to ML

- Decision tree
(Supervised Learning) :

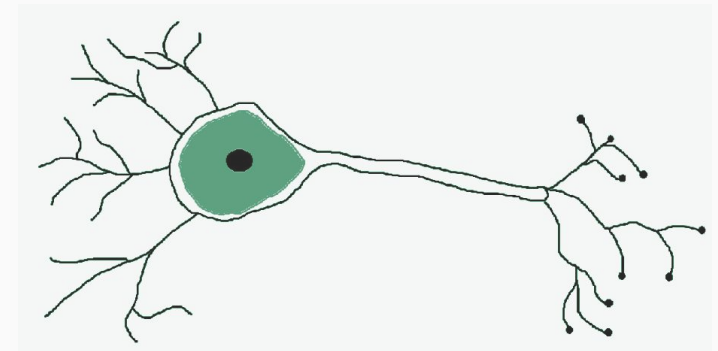
A tree and branch-based model used to map decisions and their possible consequences, similar to a flow chart.

- And So Much More ...

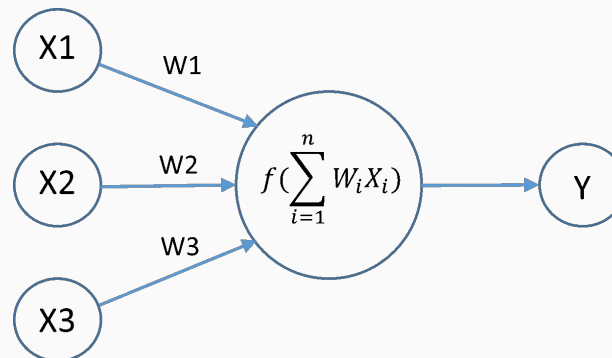
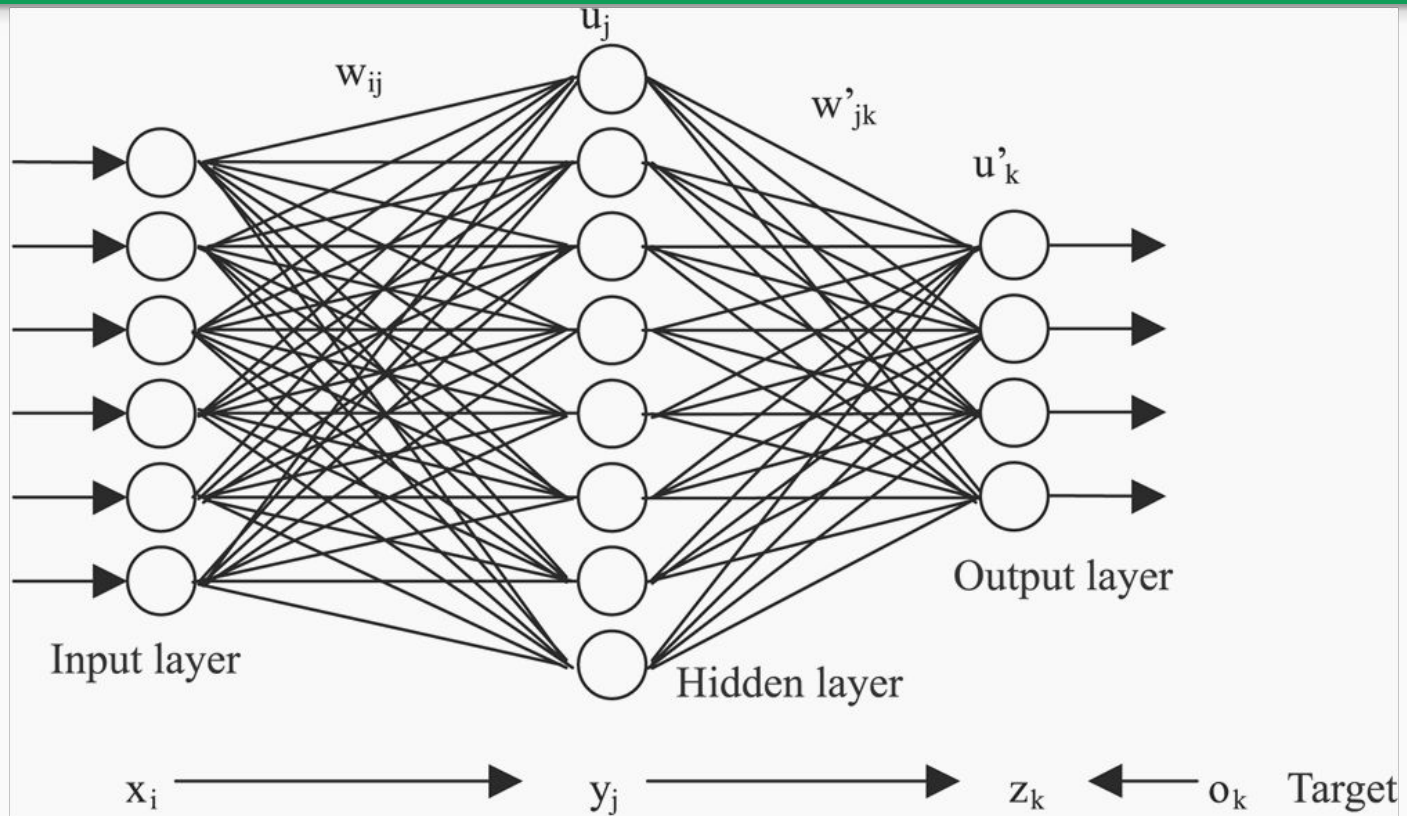


Quick Intro to ML

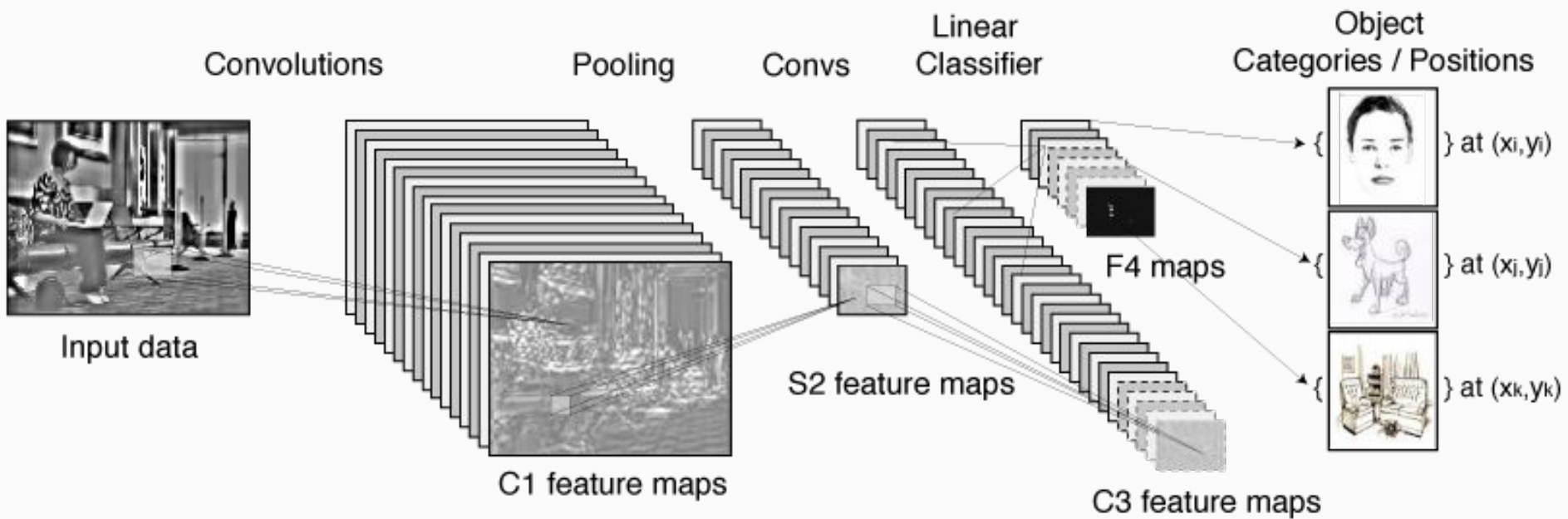
- (Artificial) Neural Networks: are computing systems vaguely inspired by the biological neural networks that constitute animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules.
- Convolutional neural network (CNN):
A type of neural networks that identifies and makes sense of images.



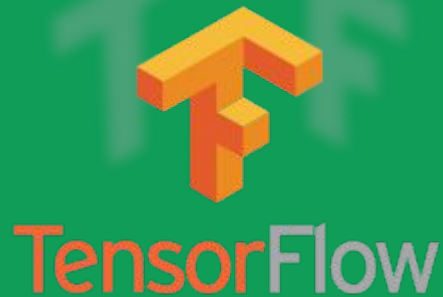
Neural Networks



Convolutional neural network



What is Tensorflow?



An open source machine learning framework for large-scale projects.

Created by the Google Brain team.

TensorFlow bundles together a slew of machine learning and deep learning (aka neural networking) models and algorithms and makes them useful by way of a common metaphor.

Biggest benefit of TensorFlow:

- Python for a convenient front-end API
- Executing in high-performance C++
- Instead of dealing with the nitty-gritty details of implementing algorithms, or figuring out proper ways to hitch the output of one function to the input of another, the developer can focus on the overall logic of the application.
- TensorFlow takes care of the details behind the scenes.

Competitors

TensorFlow competes with a slew of other machine learning frameworks. PyTorch, CNTK, and MXNet are three major frameworks that address many of the same needs.

Lets see where they stand out and come up short against TensorFlow.

PyTorch (Facebook)

- Python
- Hardware-accelerated components under the hood
- A highly interactive development model that allows for design-as-you-go work
- a better choice for fast development of projects that need to be up and running in a short time, but TensorFlow wins out for larger projects and more complex workflows.

CNTK (Microsoft Cognitive Toolkit)

- Like TensorFlow uses a graph structure to describe dataflow
- but focuses most on creating deep learning neural networks.
- handles many neural network jobs faster
- Has a broader set of APIs: Python, C++, C#, Java, ...

Apache MXNet (Adopted by Amazon)

- As the premier deep learning framework on AWS
- Can scale almost linearly across multiple GPUs and multiple machines.
- Broad range of language APIs: Python, C++, Scala, ...

What is MNIST

MNIST

- (0, 1, ... 9)
- 28x28 pixels
- 784D space
- Grayscale levels
- 70,000 images
- 60,000 training
- 10,000 testing

The MNIST database is a large database of handwritten digits that is commonly used for training various image processing systems.

A standard benchmark for neural network classification is the MNIST digits dataset, a set of 70,000 28×28 images of handwritten digits. Each MNIST digit is labeled with the correct digit class (0, 1, ... 9).

Building Model with Python Like Google

Installation Steps

- Python 3.5 or later

Built and tested TensorFlow on the following 64-bit laptop/desktop operating systems:

- macOS 10.12.6 (Sierra) or later.
- Ubuntu 16.04 or later
- Windows 7 or later.
- Raspbian 9.0 or later.


<https://www.tensorflow.org/install/>

```
C:\> pip3 install --upgrade tensorflow
```

To install the GPU version of TensorFlow, enter the following command:

```
C:\> pip3 install --upgrade tensorflow-gpu
```

1. Libraries to call



```
import os
import os.path as path

import tensorflow as tf
from tensorflow.python.tools import freeze_graph
from tensorflow.python.tools import optimize_for_inference_lib


from tensorflow.examples.tutorials.mnist import input_data
```

2. Main Program Flow

Define Model Input/Output Structure > Build Model Layer > Train > Export to file

```
def main():  
    if not path.exists('out'):  
        os.mkdir('out')  
  
    input_node_name = 'input'  
    keep_prob_node_name = 'keep_prob'  
    output_node_name = 'output'  
  
    x, keep_prob, y_ = model_input(input_node_name, keep_prob_node_name)  
  
    train_step, loss, accuracy, merged_summary_op = build_model(x, keep_prob,  
        y_, output_node_name)  
    saver = tf.train.Saver()  
  
    train(x, keep_prob, y_, train_step, loss, accuracy,  
        merged_summary_op, saver)  
  
    export_model([input_node_name, keep_prob_node_name], output_node_name)
```

3. Define Model input / output Structure



```
def model_input(input_node_name, keep_prob_node_name):  
    x = tf.placeholder(tf.float32, shape=[None, 28*28], name=input_node_name)  
    keep_prob = tf.placeholder(tf.float32, name=keep_prob_node_name)  
    y_ = tf.placeholder(tf.float32, shape=[None, 10])  
    return x, keep_prob, y_
```

4. Build up Model Layers Structure

```
def build_model(x, keep_prob, y_, output_node_name):
    x_image = tf.reshape(x, [-1, 28, 28, 1])
    # 28*28*1

    conv1 = tf.layers.conv2d(x_image, 64, 3, 1, 'same', activation=tf.nn.relu)
    # 28*28*64
    pool1 = tf.layers.max_pooling2d(conv1, 2, 2, 'same')
    # 14*14*64

    conv2 = tf.layers.conv2d(pool1, 128, 3, 1, 'same', activation=tf.nn.relu)
    # 14*14*128
    pool2 = tf.layers.max_pooling2d(conv2, 2, 2, 'same')
    # 7*7*128

    conv3 = tf.layers.conv2d(pool2, 256, 3, 1, 'same', activation=tf.nn.relu)
    # 7*7*256
    pool3 = tf.layers.max_pooling2d(conv3, 2, 2, 'same')
    # 4*4*256

    # loss
    loss = tf.reduce_mean(
        tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=logits))

    # train step
    train_step = tf.train.AdamOptimizer(1e-4).minimize(loss)

    # accuracy
    correct_prediction = tf.equal(tf.argmax(outputs, 1), tf.argmax(y_, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

    return train_step, loss, accuracy, merged_summary_op
```

5. Train the model with Images

```
def train(x, keep_prob, y_, train_step, loss, accuracy):
    mnist = input_data.read_data_sets("MNIST_data/")
    with tf.Session() as sess:
        sess.run(tf.global_variables_initializer())
        tf.train.write_graph(sess.graph_def, 'out', MODEL_NAME + '.pbtxt', True)
        for step in range(3000):
            batch = mnist.train.next_batch()
            if step % 100 == 0:
                train_accuracy = accuracy.eval(feed_dict={
                    x: batch[0], y_: batch[1], keep_prob: 1.0})
                print('step %d, training accuracy %f' % (step, train_accuracy))
            saver.save(sess, 'out/' + MODEL_NAME + '.chkp')
            test_accuracy = accuracy.eval(feed_dict={x: mnist.test.images,
                                                    y_: mnist.test.labels, keep_prob: 1.0})
            print('test%20accuracy%20%25g' % test_accuracy)
        print("training finished!")
```

English MNIST on Android With Kotlin

DEMO: MNIST Android App

- Using Kotlin Language Programming
- Tensorflow Dependency
- Tensorflow MNIST Model
- Custom Drawing Canvas 28x28
- Use Tensorflow Light Magic to detect the drawing



Installation Steps

- If you already have Android Studio, skip to Step 5!
- Select and Set correct SDK and NDK versions
(Simply search for latest versions on internet)
- Step 1 Download Android studio
<https://developer.android.com/studio/index.html>
- Step 2 Download Android SDK
\$ wget https://dl.google.com/android/android-sdk_r27.1.1-linux.tgz
\$ tar xvfz android-sdk_r27.1.1-linux.tgz -C ~/tensorflow
- Step 3 Download Android NDK
\$ wget https://dl.google.com/android/repository/android-ndk-r12b-linux-x86_64.zip
\$ unzip android-ndk-r12b-linux-x86_64.zip -d ~/tensorflow
- Step 4 Find pre-trained model OR Train Model your self
- Step 5 Get the latest Tensorflow Android release package

```
dependencies {  
    implementation 'org.tensorflow:tensorflow-android:1.10.0-rc1'  
}
```

1. User Interface

<https://raw.githubusercontent.com/miyosuda/TensorFlowAndroidMNIST/master/app/src/main/java/jp/narr/tensorflowmnist/DrawRenderer.java>

```
<io.github.yazdipour.mnist_kotlin.views.DrawView  
    android:id="@+id/drawView"  
    android:layout_width="300dp"  
    android:layout_height="300dp"  
    android:layout_gravity="center"  
    android:background="#fff" />
```

2.Loading Trained Model on a Thread

```
//creates a model object in memory using the saved tensorflow protobuf model file  
//which contains all the learned weights  
private fun loadModel() {  
    Thread(Runnable {  
        try {  
            //add our classifiers to our classifier arraylist  
            mClassifiers.add(  
                TensorFlowClassifier.create(assets, "TensorFlow",  
                    "opt_mnist_convnet-tf.pb", "labels.txt", PIXEL_WIDTH,  
                    "input", "output", true))  
        } catch (e: Exception) {  
            throw RuntimeException("Error initializing classifiers!", e)  
        }  
    }).start()  
}
```

3.OnClassifyClicked()

```
if (view.id == R.id.btn_class) {  
    //if the user clicks the classify button  
    //get the pixel data and store it in an array  
    val pixels = drawView?.getPixelData()  
  
    //init an empty string to fill with the classification output  
    var text = ""  
    //for each classifier in our array  
    for (classifier in mClassifiers) {  
        //perform classification on the image  
        val res = classifier.recognize(pixels)  
        //if it can't classify, output a question mark  
        if (res.getLabel() == null) {  
            text += classifier.name() + ": ?\n"  
        } else {  
            //else output its name  
            text += String.format("%s: %s, %f\n", classifier.name(), res.getLabel(),  
                res.getConf())  
        }  
    }  
    tfRes?.setText(text)  
}
```

4.Loading Labels

```
//given a saved drawn model, lets read all the classification labels that are  
//stored and write them to our in memory labels list  
private static List<String> readLabels(AssetManager am, String fileName) throws IOException {  
    BufferedReader br = new BufferedReader(new InputStreamReader(am.open(fileName)));  
  
    String line;  
    List<String> labels = new ArrayList<>();  
    while ((line = br.readLine()) != null) {  
        labels.add(line);  
    }  
  
    br.close();  
    return labels;  
}
```

5.Recognize Method(image) -> result

```
@Override
public Classification recognize(final float[] pixels) {

    //using the interface
    //give it the input name, raw pixels from the drawing,
    //input size
    tfHelper.feed(inputName, pixels, 1, inputSize, inputSize, 1);

    //probabilities
    if (feedKeepProb) {
        tfHelper.feed("keep_prob", new float[] { 1 });
    }
    //get the possible outputs
    tfHelper.run(outputNames);

    //get the output
    tfHelper.fetch(outputName, output);

    // Find the best classification
    //for each output prediction
    //if its above the threshold for accuracy we predefined
    //write it out to the view
    Classification ans = new Classification();
    for (int i = 0; i < output.length; ++i) {
        System.out.println(output[i]);
        System.out.println(labels.get(i));
        if (output[i] > THRESHOLD && output[i] > ans.getConf()) {
            ans.update(output[i], labels.get(i));
        }
    }

    return ans;
}
```

Building a Persian MNIST Model Using CustomVision.Ai



The **Custom Vision**

Service is a Microsoft Cognitive Service that lets you build custom image classifiers. It makes it easy and fast to build, deploy, and improve an image classifier.

CustomVision

- Part of Azure Cloud Services
- Free and Paid Services
- REST API
- [Training API](#)
- [Prediction API](#)
- A web interface to upload your images and train the classifier.
- 50 images per class are enough to start your prototype.
- Model export to ONNX ,Dockerfile, TensorFlow, CoreML
- SDK and Complete sample and Tutorial Videos for cross platform scenarios

Making a model

Custom Vision

Projects

1 New Project

New project

Name*
eShopOnContainersAI 1

Description
Enter project description

Resource Group* [create new](#)
Limited trial

Project Types ⓘ
☒ Classification 2
☐ Object Detection (preview)

Domains ⓘ
☐ General
☐ Food
☐ Landmarks
☐ Retail
☐ Adult
☒ General (compact) 3
☐ Landmarks (compact)
☒ Retail (compact)

Cancel Create project 4

Making a model

mnist_fa

Training Images

Perform

Filter

Iteration

Workspace

Tags

Tagged

Untagged

Showing: all tagged images

Search for

☐ 0 5 ...

☐ 1 5 ...

☐ 2 5 ...

☐ 3 6 ...

☐ 4 5 ...

☐ 5 5 ...

☐ 6 5 ...

☐ 7 8 ...

☐ 8 5 ...


☐ 9 5 ...

Add images


Delete

Tag images


Select all



Making a model

 mnist_fa

Iterations

Probability Threshold: 50% 

Iteration 15

Trained : moments ago with General (compact) domain

Iteration 14


Trained : 2 days ago with General (compact) domain


Iteration 13


Trained : 2 days ago with General (compact) domain


Iteration 12

Trained : 2 days ago with General (compact) domain

 Prediction URL


 Make default


 Delete


 Export

Iteration 15


Finished training on 8/1/2018 8:49:19 PM using General (compact) domain
Classification type: Multiclass (Single tag per image)

Precision 

Recall 



70.7%




70.7%

Performance Per Tag

Tag	Precision	Recall
1	100.0%	72.2%
0	100.0%	75.9%
8	87.8%	82.1%

Making a model

 mnist_fa

Iterations

Probability Threshold: 50% ⓘ

Iteration 14

Trained : 24 minutes ago
with General (compact)
domain

Iteration 13

Trained : 26 minutes ago
with General (compact)
domain

Iteration 12

Trained : 28 minutes ago
with General (compact)
domain

Iteration 11

Prediction URL ✓ Already Default

Iteration 14

Finished training on 7/30/2018 1:18:11 PM using
Classification type: Multiclass (Single tag per im

Precision ⓘ


63.5%


Performance Per Tag


Tag

1

Choose your platform


CoreML
iOS 11


TensorFlow
Android


ONNX
Windows ML

A Database of Persian HSF from Prof. Javad Sadri (Concordia University)

A novel comprehensive database for offline Persian handwriting recognition

<https://www.sciencedirect.com/science/article/pii/S0031320316300097>

Comprehensive Database for Offline Persian Handwritten Recognition

https://users.encs.concordia.ca/~j_sadri/PersianDatabase.htm

- 250 men+250 women
- 7 different pages of handwritten sample forms (HSFs).
- 3 color formats: true color, gray level, and binary
- 250 Digit Sample Available for Free

Item	Total	Training Set	Validation Set	Testing Set
Dates	4500	2700	1800	1800
Digits	97124	60000	17124	20000
Numerical Strings	19500	11700	3900	3900
Alphabet Letters	43000	25800	8600	8600
Punctuation and Symbols	16000	9600	3200	3200
Words	70000	42000	14000	14000
Texts	500	300	100	100
Writer	500 (250 Male+250 Female)	-----	-----	-----
HSF *	3500	-----	-----	-----

* HSF stands for Handwritten Sample Form

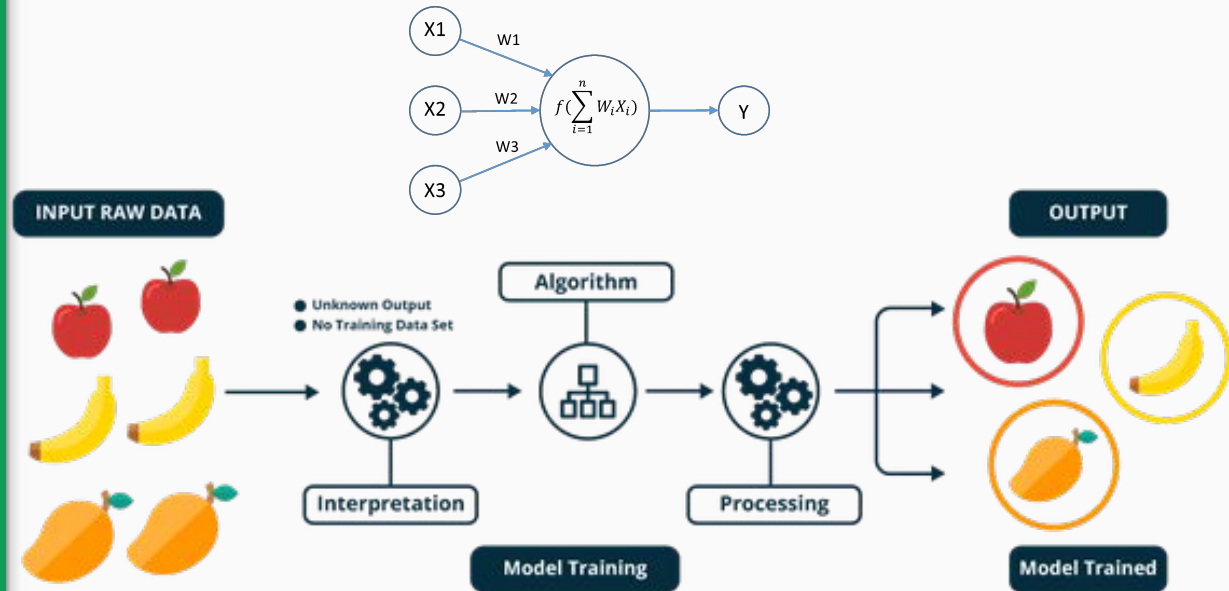
Models

Models

With your own data, you can train custom models to perform tasks like recognizing images, extracting meaning from text, or finding relationships between numerical values.

Pre-trained models are deep learning model weights that you can download and use without training.

Simply give some input and get a result.



Popular Models

Tensorflow Model (pb):

Platforms: Cloud, Android, IoT

Frameworks: Mostly Every ML framework

Big Difference: The .pb file always comes with a label.txt file

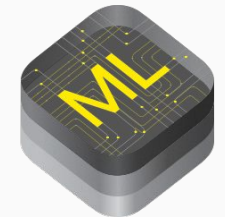
* Easy to find open source models for specific needs.



CoreML Model (mlmodel):

Platforms: iOS >11, macOS

Frameworks: CoreML, Caffe, Keras



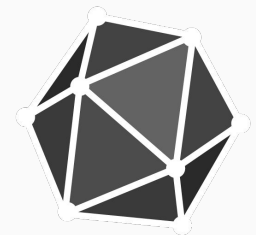
Open Neural Network Exchange (onnx):

Platforms: Cloud, WinML, IoT, Android (Xamarin)

Frameworks: Mostly Every ML framework

* Same as CoreML but more open and convertible

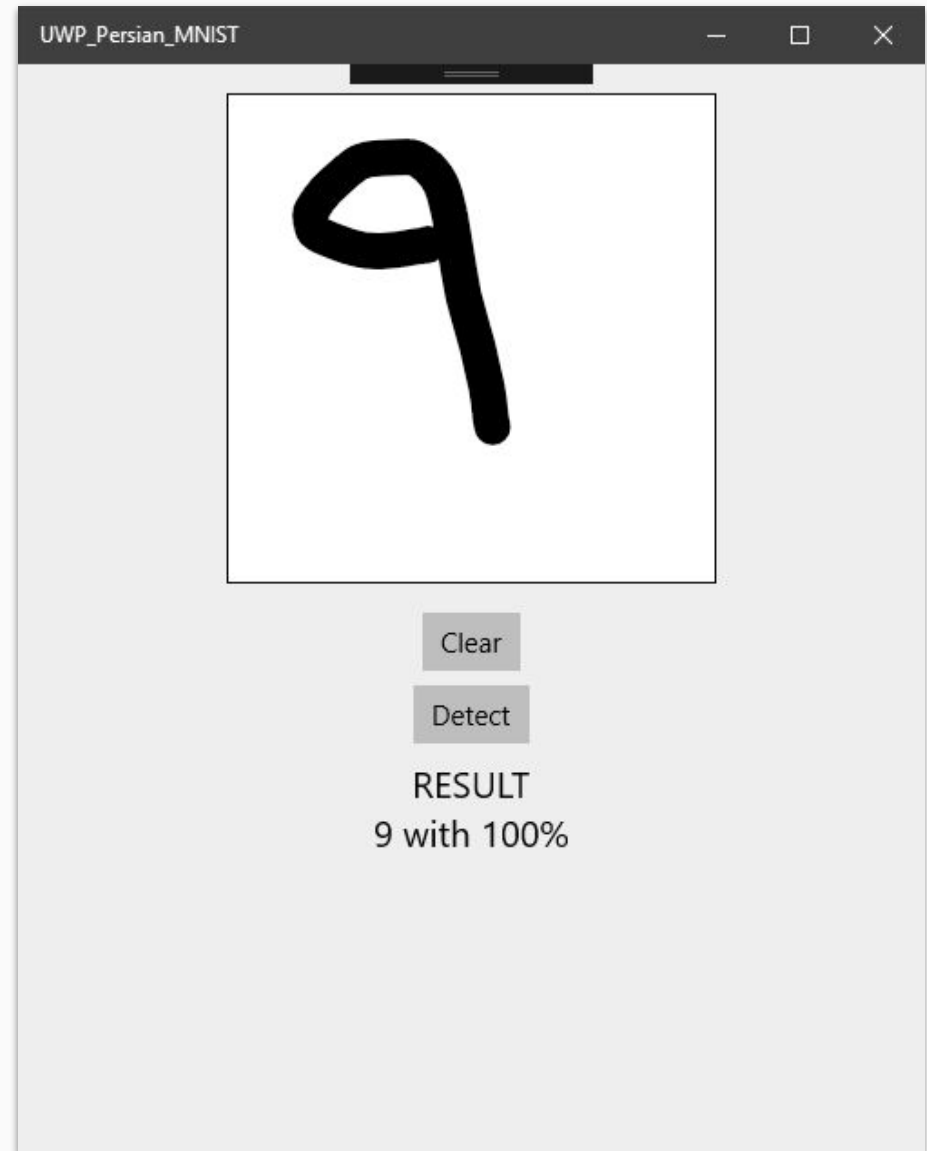
* Easy to find open source models for specific needs.



ML On All Platforms

UWP Example

- Using a Ink Canvas
- Using CustomVision REST API with its prediction key
- Make a POST Request with your Image in the request BODY
- Parse JSON result



1. User Interface

```
<StackPanel HorizontalAlignment="Center">
    <Border Background="White" BorderThickness="1" Width="270" Height="270" BorderBrush="Black">
        <InkCanvas x:Name="canvas" Width="270" Height="270" />
    </Border>
    <Button HorizontalAlignment="Center" Content="Clear" Click="Clear_Click" />
    <Button HorizontalAlignment="Center" Content="Detect" Margin="0,8,0,8" Click="Detect_Click" />
    <TextBlock HorizontalAlignment="Center" TextAlignment="Center" TextWrapping="Wrap"
        Style="{ThemeResource SubtitleTextBlockStyle}" Text="RESULT" />
    <TextBlock HorizontalAlignment="Center" x:Name="textBlock" TextAlignment="Center"
        Style="{ThemeResource SubtitleTextBlockStyle}" />
</StackPanel>
```

2. Save Canvas

```
private async Task Save()
{
    StorageFolder storageFolder = KnownFolders.PicturesLibrary;
    var file = await storageFolder.CreateFileAsync(fileName,
        CreationCollisionOption.ReplaceExisting);

    CanvasDevice device = CanvasDevice.GetSharedDevice();
    CanvasRenderTarget renderTarget = new CanvasRenderTarget(device,
        (int)canvas.ActualWidth, (int)canvas.ActualHeight, 10);
    using (var ds = renderTarget.CreateDrawingSession())
    {
        ds.Clear(Colors.White);
        ds.DrawInk(canvas.InkPresenter.StrokeContainer.GetStrokes());
    }
    using (var fileStream = await file.OpenAsync(FileAccessMode.ReadWrite))
    {
        await renderTarget.SaveAsync(fileStream, CanvasBitmapFileFormat.Bmp, 1f);
    }
}
```

3. Read Canvas + HttpRequest to CV Server

```
private async void Read()
{
    textBlock.Text = "";
    try
    {
        StorageFile file = await KnownFolders.PicturesLibrary.GetFileAsync(fileName);
        using (var stream = await file.OpenReadAsync())
        {
            var client = new HttpClient();
            client.DefaultRequestHeaders.Add("Prediction-Key", "38a82ee28b694cd4ab3678e2041");
            using (var content = new StreamContent(stream.AsStream()))
            {
                content.Headers.ContentType = new MediaTypeHeaderValue("application/octet-stream");
                HttpResponseMessage response = await client.PostAsync(url, content);
                var result = await response.Content.ReadAsStringAsync();
                var model = Newtonsoft.Json.JsonConvert.DeserializeObject<CVModel>(result);
                if (model.Predictions == null) throw new Exception("Error");
                List<Prediction> predictions = new List<Prediction>(model.Predictions);
                predictions = predictions.OrderBy(o => o.Probability).ToList();
                textBlock.Text = $"{model.Predictions[0].tagName} with {model.Predictions[0].Probability*100}%";
            }
        }
    }
    catch (Exception ex)
    {
        textBlock.Text = ex.Message;
    }
}
```

4. Get and Analyse JSON result

```
{
  "Id": "3f76364c-b8ae-4818-a2b2-2794cfbe377a",
  "Project": "2277aca4-7aff-4742-8afb-3682e251c913",
  "Iteration": "84105bfe-73b5-4fcc-addb-756c0de17df2",
  "Created": "2018-05-03T14:15:22.5659829Z",
  "Predictions": [
    {"TagId": "35ac2ad0-e3ef-4e60-b81f-052a1057a1ca", "Tag": "1", "Probability": 0.102716163},
    {"TagId": "28e1a872-3776-434c-8cf0-b612dd1a953c", "Tag": "2", "Probability": 0.02037274},
    {"TagId": "28e1a872-3776-434c-8cf0-b612dd1a953c", "Tag": "3", "Probability": 0.01037274},
    {"TagId": "28e1a872-3776-434c-8cf0-b612dd1a953c", "Tag": "4", "Probability": 0.00037274}
  ]
}
```

“The Future of Cloud Computing Will Blow Your Mind”

- Everywhere
- ML Everywhere
- ML-as-a-Service
- Always Update ML Models
- Distributed Computing
- Power of Quantum Computing

Thanks!

Contact:

📄 <https://yazdipour.github.io/>

✉ yazdipour@outlook.com

Source Code:

https://github.com/yazdipour/Simple_Persian_MNIST

References:

- Wikipedia
- https://www.tensorflow.org/versions/r1.0/get_started/mnist/beginners
- <https://www.infpworld.com/article/3278008/machine-learning/what-is-tensorflow-the-machine-learning-library-explained.html>
- <https://dzone.com/articles/ai-glossary>
- <https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/home>