Microsoft

Evaluación de los requisitos para implementar las API de Custom Vision

BIENVENIDA

Evalúe los requisitos de una solución que implementa Custom Vision Prediction API y Training API. Los planes de diseño completados serán compatibles con los requisitos del flujo de trabajo, y estará mejor preparado para trabajar con desarrolladores y arquitectos.

En este módulo aprenderá a:

- Evaluar los requisitos de la autorización del servicio
- Examinar la API para la predicción de imagen
- Examinar la API para el entrenamiento de los modelos predictivos

Requisitos previos

- Entender los conceptos de la nube
- Conocer los servicios principales de Azure
- Conocer las funcionalidades de seguridad, privacidad, cumplimiento y confianza
- Conocer el soporte técnico y los precios de Azure

Introducción 4 min

API de Custom Vision 6 min

Investigación de la autorización del servicio 5 min

Examinar Custom Vision Prediction API 10 min

Examinar Custom Vision Training API 10 min

Prueba de conocimientos 8 min

Resumen 3 min

3

INTRODUCCIÓN

Imagine que dirige un negocio de embalaje que distribuye productos por todo el mundo. Su empresa ha realizado una fuerte inversión en la automatización del empaquetado de las cajas y la optimización del tamaño de cada caja enviada. Pero ha recibido algunas quejas sobre embalajes rasgados o dañados. Los miembros del equipo de seguridad de productos están valorando cómo inspeccionar productos empaquetados antes del envío. Quieren automatizar el proceso de inspección e implicar solo a una persona si se determina que el embalaje es defectuoso.

El servicio cognitivo Azure Custom Vision combina la inteligencia artificial y el aprendizaje automático para ofrecer un servicio sofisticado de detección y clasificación de imágenes.

Estas capacidades se pueden usar directamente desde el portal web del servicio Custom Vision para crear y entrenar un modelo, probar su precisión y, después, usarlo para evaluar y aplicar etiquetas a las imágenes. Pero este proceso no es suficiente para su caso de uso.

Por ello, en su lugar, el equipo de desarrollo quiere utilizar las API REST que permiten a las aplicaciones y los scripts personalizados aprovechar las mismas capacidades del servicio Custom Vision que se

proporcionan en el portal web. Con las API de Custom Vision, su empresa puede integrar este servicio cognitivo directamente en el proceso de automatización existente para identificar de forma rápida y precisa los paquetes defectuosos antes de que se envíen a los clientes.

En este módulo, aprenderá a:

- Evaluar los requisitos de la autorización del servicio
- Examinar la API para la predicción de imagen
- Examinar la API para el entrenamiento de los modelos predictivos

API DE CUSTOM VISION

Las API de Custom Vision usan algoritmos integrados que proporcionan los servicios con "inteligencia". Puede usar estos algoritmos para crear y personalizar sus propios modelos de Computer Vision de última generación con solo unas pocas líneas de código.

Los algoritmos de Custom Vision API se exponen como llamadas de servicio simples basadas en REST. Los desarrolladores pueden usar las llamadas de REST para crear proyectos (clasificadores), cargar y etiquetar imágenes, realizar el entrenamiento, administrar iteraciones del modelo e incluso realizar predicciones.

Sugerencia

Este módulo se centra en los servicios REST que se pueden usar en cualquier lenguaje. Microsoft también publica SDK de código abierto para .NET, Python, Java, Node.js, y Go que encapsulan las API REST

subyacentes en objetos y métodos específicos de la plataforma. Consulte la documentación para obtener más información sobre los SDK disponibles.

API de Custom Vision

Las API de Custom Vision son una manera perfecta de usar la potencia de la inteligencia artificial para:

- Crear modelos de Computer Vision de última generación.
- Cargar y etiquetar imágenes de entrenamiento.
- Entrenar clasificadores para el aprendizaje activo.
- Realizar predicción de imágenes para identificar las coincidencias probables con un modelo entrenado.
- Realizar detección de objetos para buscar elementos *dentro* de una imagen y devolver un rectángulo de selección.

En general, los métodos proporcionados por las API de Custom Vision se dividen en las siguientes categorías:

- **Entrenamiento**. Administre proyectos (clasificadores) y cargue etiquete y administre las imágenes de entrenamiento.
- **Predicción**. Realice predicciones basadas en los modelos de clasificación.

Al igual que con la Computer Vision API relacionada, para llamar a las API de Custom Vision, solo tiene que hacer una solicitud web segura. Por ejemplo, aquí se muestra una versión simplificada del aspecto que tendría una llamada a una Custom Vision API en Python:

Python

```
conn =
httplib.HTTPSConnection('southcentralus.api.cognitive.microsoft.com')

conn.request("POST", serviceEndpointUrl, body)

response = conn.getresponse()


data = response.read()
```

La misma llamada API en C# es similar:

C#Copiar

```
HttpClient client = new HttpClient();

StringContent content = new StringContent(body);

response = await client.PostAsync(serviceEndpointUrl, content);


string data = await response.Content.ReadAsStringAsync();
```

INVESTIGACIÓN DE LA AUTORIZACIÓN DEL SERVICIO

Las API de Custom Vision usan un par de *claves de suscripción* para controlar el acceso a las dos API del servicio:

- **La clave de entrenamiento** para acceder a los miembros de la API utilizados a fin de entrenar el modelo.
- **La clave de predicción** para acceder a los miembros de la API utilizados para clasificar las imágenes con un modelo entrenado.

La separación de las claves de suscripción de Custom Vision API facilita la entrega de las claves en función de los requisitos técnicos, de la aplicación o del negocio. Por ejemplo, un servicio que solo requiere tareas de predicción solo podrá usar esa API, y solo se le cobrará por la cuota de predicción. Ambas claves están disponibles en el portal al crear la instancia del servicio Custom Vision.

Sugerencia

Las claves también pueden encontrarse en la página de Azure Portal correspondiente al recurso de Azure de Custom Vision asociado con el proyecto, en la hoja **Claves**.

Suministro de una clave de suscripción

Cada llamada mediante programación a las API de Custom Vision requiere que se pase una de estas claves de suscripción al servicio como un valor de encabezado de solicitud. Por ejemplo, si el código llamara a Prediction API en C#, usaría la colección DefaultRequestHeaders para agregar la clave de predicción a la solicitud con la clave Prediction-Key:

C#Copiar

```
string predictionKey = "...";

HttpClient client = new HttpClient();

client.DefaultRequestHeaders.Add("Prediction-Key", predictionKey)
```

El mismo código en Ruby tendría un aspecto similar a este ejemplo:

RubyCopiar

```
request['Prediction-Key'] = '{prediction key}'
```

Dado que las API de Custom Vision se dividen en distintas API de predicción y entrenamiento, echemos un vistazo a cada API con más detalle, empezando por Custom Vision Prediction.

EXAMINAR CUSTOM VISION PREDICTION API

La tarea más común que realizará una aplicación o un script con el servicio Custom Vision es solicitar predicciones de imágenes con Prediction API. Esta tarea implica el envío de una solicitud web autorizada al punto de conexión de la suscripción y el procesamiento de la información devuelta por la llamada.

Sugerencia

Con fines de prueba, los suscriptores pueden practicar con el uso de Prediction API mediante la **consola de pruebas de Custom Vision Prediction API**. Asegúrese de seleccionar la consola de pruebas asociada a la región en la que se implementa el servicio.

Llamada a Prediction API

A Prediction API se accede a través del punto de conexión web público específico del servicio registrado de Custom Vision. Hay dos API disponibles en función del tipo de proyecto: clasificación de imágenes y detección de objetos.

Clasificación de imágenes

La clasificación de imágenes analiza una imagen suministrada y devuelve una lista de etiquetas identificadas en la imagen. Se proporcionan varios puntos de conexión. Aquí se muestran los dos puntos de conexión que se usan con más frecuencia:

- **ClassifyImage** acepta una carga útil binaria ("application/octet-stream") como una matriz de bytes que contiene los datos de la imagen.
- **ClassifyImageUrl** acepta una carga útil de JSON ("application/json") que especifica una dirección URL de la imagen disponible públicamente.

Detección de objetos

La detección de objetos es similar a la clasificación de imágenes, pero también devuelve las coordenadas (rectángulo de selección) de dónde se ha ubicado la etiqueta dentro de la imagen. Al igual que con la clasificación, hay dos puntos de conexión de uso frecuente disponibles:

- **DetectImage** acepta una carga útil binaria ("application/octet-stream") como una matriz de bytes que contiene los datos de la imagen.
- **DetectImageUrl** acepta una carga útil de JSON ("application/ejson") que especifica una dirección URL de la imagen disponible públicamente.

Ambos puntos de conexión de la API usan la misma estructura para formar la dirección URL:

text

https://{endpoint}/customvision/v3.0/Prediction/{projectId}/{projectType}/iterations/{iterationName}/{imageType}

En esta dirección URL:

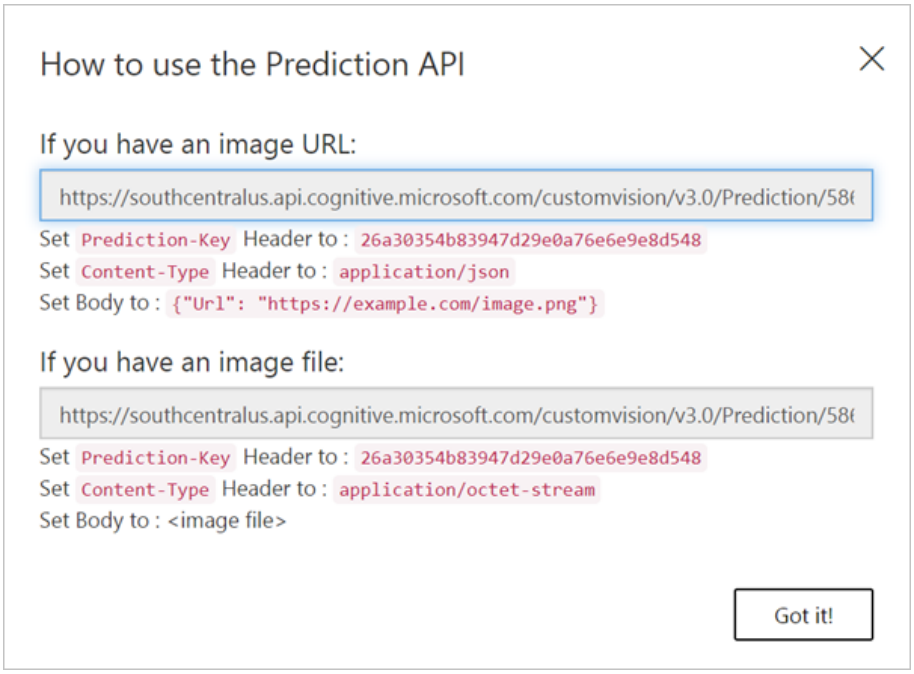
- {endpoint} es el punto de conexión de ubicación en el que se ha creado el servicio, por ejemplo, southcentralus.api.cognitive.microsoft.com.
- {projectId} es un identificador de proyecto único que se usa para especificar el servicio de Custom Vision.
- {projectType} es classify para la clasificación de imágenes o detect para la detección de objetos.
- {iterationName} es el nombre de la iteración del modelo entrenado que se usa.
- {imageType} es url cuando la imagen se pasa como una dirección URL o image cuando la imagen se pasa como datos binarios en el cuerpo de la solicitud.

Importante

La API a la que se llama, Detect frente a Classify, se basa en el *tipo* de proyecto creado en el servicio Custom Vision. Si no llama a la API adecuada, el servicio devolverá 400 Bad Request - Invalid project type for operation.

Búsqueda de la dirección URL del proyecto

Los detalles específicos del punto de conexión están disponibles en el portal del servicio Custom Vision. Si selecciona el botón **Ver punto de conexión** en la pestaña **Predicciones** o la opción **Dirección URL de Prediction** en la pestaña **Rendimiento**, se mostrará un cuadro de diálogo similar al siguiente:



Construcción de la solicitud HTTP

Después de identificar la dirección URL correcta, se invoca mediante una solicitud HTTP PUT. Recuerde que es necesario pasar una clave de predicción con la solicitud. Esta llave se proporciona como un encabezado de solicitud con el nombre Prediction-Key.

Por ejemplo, el código de C# siguiente envía una dirección URL de la imagen a Prediction API con el punto de conexión y la clave de predicción especificados:

C#

```
public async Task<string> MakePredictionRequestAsync(string url, string predictionKey, string imageUrl)

{

    var client = new HttpClient();
```

```
    client.DefaultRequestHeaders.Add("Prediction-Key", predictionKey);

    using (var content = new StringContent("{\"Url\":\"" + imageUrl + "\"}", Encoding.UTF8, "application/json");)

    {

        var response = await client.PostAsync(url, content);

        return await response.Content.ReadAsStringAsync();

    }

}
```

En Python 3, el mismo código podría ser similar al de este ejemplo:

PythonCopiar

```
import http.client, urllib.request, urllib.parse, urllib.error, base64


headers = {

    'Content-Type': 'application/json',

    'Prediction-Key': '{prediction key}',

}


body = "{'Url' : '" + url + "' }"
```

```
try:
    conn = http.client.HTTPSConnection('southcentralus.api.cognitive.microsoft.com')
    conn.request("POST", "/customvision/v3.0/Prediction/{projectId}/classify/iterations/{iterationName}/url", body, headers)
    response = conn.getresponse()
    data = response.read()
    print(data)
    conn.close()
except Exception as e:
    print("[Errno {0}] {1}".format(e.errno, e.strerror))
```


Nota

Los métodos de Prediction API también aceptan un parámetro de cadena de consulta opcional llamado application para identificar la aplicación que invoca el servicio. Este valor se puede utilizar para realizar el seguimiento de las aplicaciones que utilizan el servicio

Custom Vision.

Pasar archivos de imagen directamente

Si va a usar la *carga de archivo binario* **ClassifyImage** o **DetectImage**, el cuerpo de la solicitud debe contener los datos de la imagen codificados en una matriz de bytes. Esta es la misma llamada que usa el punto de conexión **ClassifyImage** en su lugar. En este caso, el encabezado Content-Type debe establecerse en "application/octet-stream".

C#

```
public async Task<string> MakePredictionRequestAsync(string url, string predictionKey, string imageFile)

{

    var fileStream = new FileStream(imageFile, FileMode.Open, FileAccess.Read);

    var binaryReader = new BinaryReader(fileStream);

    var bytes = binaryReader.ReadBytes((int)fileStream.Length);



    var client = new HttpClient();

    client.DefaultRequestHeaders.Add("Prediction-Key", predictionKey);

    using (var content = new ByteArrayContent(bytes))

    {

        content.Headers.ContentType = new MediaTypeHeaderValue("application/octet-stream");
```

```
        var response = await client.PostAsync(url, content);

        return await response.Content.ReadAsStringAsync();

    }

}
```

Procesamiento de la respuesta

Prediction API devuelve un objeto JSON que incluye varios bits de información. La información más importante aquí es una matriz de *predicciones* que indica la probabilidad de que la imagen proporcionada contuviera ese elemento entrenado. Por ejemplo, si el modelo se ha entrenado para reconocer los tipos de animales que se encuentran en el Círculo Polar Ártico y se proporciona una imagen de un oso polar, **ClassifyImage** podría devolver un objeto como el siguiente:

JSON

```
{
  "id":"71b145f7-8533-4cf4-9821-f31a4293508a",
  "project":"586008f5-2d8e-4e4a-abb4-66127e96a4f4",
  "iteration":"f118e60f-5e25-40ad-89a2-3b6b1cdd5dac",
  "created":"2019-06-19T20:55:10.240Z",
  "predictions":[
    {
```

```
      "probability":1.0,
      "tagId":"8b2ada4d-1d0f-4b07-af19-dfd643043e1f",
      "tagName":"Polar bear"
    },
    {
      "probability":3.3948698E-11,
      "tagId":"56fe8dca-2849-454a-8aa0-5897b1c30009",
      "tagName":"Arctic fox"
    },
    {
      "probability":1.96249064E-13,
      "tagId":"ae6f7ccb-46ac-44af-9b70-8d91f4951122",
      "tagName":"walrus"
    }
  ]
}
```


Cada predicción contiene un elemento tagId que identifica la etiqueta coincidente, un atributo tag que proporciona un nombre descriptivo de la etiqueta y un valor probability de 0 a 1 que indica el nivel de

confianza del servicio en la identificación de la etiqueta especificada en la imagen.

EXAMINAR CUSTOM VISION TRAINING API

El portal web del servicio Custom Vision es una forma sencilla de entrenar un modelo mediante la carga de imágenes etiquetadas. Este enfoque es la forma más común de entrenar, probar y publicar un modelo. Sin embargo, a veces una necesidad empresarial podría requerir que un modelo se prepare o se vuelva a entrenar en función de los datos entrantes disponibles para las aplicaciones que usan el servicio. En estos casos, la aplicación puede usar Training API para agregar y etiquetar nuevas imágenes y publicar una nueva iteración del servicio Custom Vision.

De forma similar a Prediction API, Training API proporciona métodos HTTP para agregar imágenes de entrenamiento en un proyecto y etiquetarlas. El proceso de llamada y uso de la respuesta es idéntico al de Prediction API. Simplemente usa puntos de conexión diferentes.

CreateImages es el método subyacente de Custom Vision que se usa para enviar imágenes de entrenamiento etiquetadas. Al igual que Prediction API, Custom Vision Training API proporciona métodos independientes para cargar archivos binarios y proporcionar direcciones URL de imagen disponibles públicamente:

- **CreateImagesFromFiles** incluye uno o varios archivos de imagen codificados y etiquetas opcionales para crear imágenes. Hay un límite de 64 imágenes y 20 etiquetas.

- **CreateImagesFromUrls** especifica una o varias direcciones URL y etiquetas opcionales para crear imágenes. Hay un límite de 64 imágenes y 20 etiquetas.

Nota

Hay varios métodos adicionales disponibles en la Training API que permiten crear y buscar proyectos, crear y eliminar etiquetas, entrenar proyectos y mucho más. Todo lo que puede hacer en el portal, normalmente puede hacerlo con la API REST.

Todos los puntos de conexión tienen el mismo formato básico que los de Prediction API:

text

https://{endpoint}/customvision/v3.0/training/projects/{projectId}/images/{imageType}

En esta dirección URL:

- {endpoint} es el punto de conexión de ubicación en el que se ha creado el servicio, por ejemplo, southcentralus.api.cognitive.microsoft.com.
- {projectId} es un identificador de proyecto único que se usa para especificar el servicio de Custom Vision.
- {imageType} se corresponde con urls, cuando las imágenes se pasan como direcciones URL o con files cuando las imágenes se pasan como datos codificados en el cuerpo de la solicitud.

Compilación de una solicitud

El punto de conexión de Training API está disponible en el panel **Configuración** del proyecto del servicio Custom Vision en el portal web. También puede encontrar la *clave de entrenamiento* en esta página. Necesita la clave de entrenamiento para autorizar llamadas a los servicios de Training API. Una vez que tenga estos dos fragmentos de información, está listo para usar los métodos **CreateImages**.

Una vez identificada la dirección URL correcta, se invoca con una solicitud HTTP PUT que pasa la solicitud en el cuerpo y la clave de entrenamiento como un encabezado de solicitud con el nombre Training-Key.

Puede usar la mayoría de los tipos de elementos multimedia disponibles para estructurar el cuerpo de la solicitud: "application/json", "application/xml", "text/xml" o "applicatio n/x-www-form-urlencoded". Puede elegir el tipo con el que le resulte más fácil trabajar en su lenguaje o marco. El valor de encabezado Content-Type determinará el tipo de carga útil del contenido que proporcione.

Cada solicitud **CreateImages** toma una lista de imágenes o direcciones URL y un conjunto opcional de etiquetas. Este es un ejemplo de un cuerpo de solicitud válido para **CreateImagesFromUrls** (en JSON):

JSON

```
{
  "images": [
```

```
    {
      "url": "{url to image #1}",
      "tagIds": [ "{tag-id}" ]
    }
  ],
  "tagIds": [
    "{tag-id}"
  ]
}
```


Las colecciones tagIds son opcionales. Puede obviarlas si no las está usando. Puede proporcionar etiquetas por imagen (si son diferentes para cada imagen) o para toda la colección de imágenes con la segunda colección de etiquetas. Debe proporcionar el identificador único para la etiqueta. Puede recuperar este identificador desde el portal web o mediante HTTP GET para invocar el método {endpoint}/customvision/v3.0/training/projects/{projectId}/tags. Este es un ejemplo de respuesta:

JSON

```
[
  {
    "id":"56fe8dca-2849-454a-8aa0-5897b1c30009",
```

```
    "name":"Arctic fox",
    "description":null,
    "type":"Regular",
    "imageCount":130
  },
  {
    "id":"ae6f7ccb-46ac-44af-9b70-8d91f4951122",
    "name":"Walrus",
    "description":null,
    "type":"Regular",
    "imageCount":138
  },
  {
    "id":"8b2ada4d-1d0f-4b07-af19-dfd643043e1f",
    "name":"Polar bear",
    "description":null,
    "type":"Regular",
    "imageCount":141
```

```
  }
]
```

Llamada al servicio

Después de compilar la solicitud, puede usar un método HTTP PUT para invocar la API. Este es un ejemplo en C#:

C#

```
public async Task<string> AddTrainingImageAsync(string url, string trainingKey, string[] imageUrls, string[] tags)
{
    string jsonData = ...; // Build the JSON request from imageUrls and tags.

    var client = new HttpClient();
    client.DefaultRequestHeaders.Add("Training-Key", trainingKey);
    using (var content = new StringContent(jsonData, Encoding.UTF8, "application/json");)
    {
        var response = await client.PostAsync(url, content);
        return await response.Content.ReadAsStringAsync();
    }
```

```
}
```

En Python, el código debería tener un aspecto parecido al siguiente:

PythonCopiar

```
import http.client, urllib.request, urllib.parse, urllib.error, base64


headers = {
    'Content-Type': 'application/json',
    'Training-Key': '{training key}',
}

body = "{json-data}"


try:
    conn = http.client.HTTPSConnection("{endpoint}")
    conn.request("POST",
"/customvision/v3.0/training/projects/{projectId}/images/urls",    body,
headers)
    response = conn.getresponse()
    data = response.read()
```

```
    print(data)

    conn.close()

except Exception as e:

    print("[Errno {0}] {1}".format(e.errno, e.strerror))
```

Procesamiento del resultado

La respuesta procedente del servicio tendrá el mismo formato que la solicitud. (Puede influir en este comportamiento mediante el valor de encabezado Accepts). El objeto devuelto tendrá un resultado general (isBatchSuccessful) y una entrada para cada imagen pasada, a fin de indicar si se ha procesado. Este es un resultado de ejemplo del método CreateImagesFromUrls:

JSON

```
{
  "isBatchSuccessful": true,
  "images": [
    {
      "sourceUrl":
"https://timedotcom.files.wordpress.com/2019/06/polar-bear-in-siberia.jpg",
      "status": "OK",
      "image": {
```

```
        "id": "955651c9-0ba3-40a0-b33f-a46d39feb3ee",

        "created": "2019-06-19T23:02:39.8757957",

        "width": 1555,

        "height": 900,

        "resizedImageUri":
"https://irisscuprodstore.blob.core.windows.net/i-
586008f52d8e4e4aabb466127e96a4f4/i-
955651c90ba340a0b33fa46d39feb3ee?sv=2017-04-
17&sr=b&sig=JfzNzS3%2B1qKKHLnfDK%2Fcuh5FDPFPxRSHZsSnz%2
Bri4II%3D&se=2019-06-20T23%3A02%3A39Z&sp=r",

        "thumbnailUri":
"https://irisscuprodstore.blob.core.windows.net/i-
586008f52d8e4e4aabb466127e96a4f4/t-
955651c90ba340a0b33fa46d39feb3ee?sv=2017-04-
17&sr=b&sig=3udivyZu%2BGtgT5%2BqX6iCDTYT8TxvBiAzKk2WJV3d
FzU%3D&se=2019-06-20T23%3A02%3A39Z&sp=r",

        "originalImageUri":
"https://irisscuprodstore.blob.core.windows.net/i-
586008f52d8e4e4aabb466127e96a4f4/o-
955651c90ba340a0b33fa46d39feb3ee?sv=2017-04-
17&sr=b&sig=bWJjIM5aBcAdx3LS5EsZGLdRmK5B%2BE%2BShzSbJD7
LZLU%3D&se=2019-06-20T23%3A02%3A39Z&sp=r",

        "tags": [

         {
```

```
            "tagId": "8b2ada4d-1d0f-4b07-af19-dfd643043e1f",
            "tagName": null,
            "created": "2019-06-19T23:02:39.9019664"
          }
        ]
      }
    }
  ]
}
```