#cheat sheet

Integrating Azure Custom Vision with Power Apps

1. Create and Train a Custom Vision Model in Azure

Step 1: Set up a Custom Vision Project

- Log in to the Azure Portal.
- Navigate to Custom Vision and create a Custom Vision resource (both Training and Prediction).
- Sign in to the Custom Vision website using your Azure credentials.
- Create a new project:
  - Assign a project name and description.
  - Choose a domain (e.g., General, Retail, Food).
  - Set a classification type: Multiclass (one tag per image) or Multilabel (multiple tags per image).

Step 2: Upload and Tag Images

- Upload images for training and group them by category (tags).
- Ensure images are of high quality and varied (at least 15-20 per tag for better accuracy).
- Tag each image accurately with its category.

Step 3: Train the Model

- Click on Train in the Custom Vision portal.
- Choose Quick Training for faster results or Advanced Training for more customized parameters.
- Review evaluation metrics such as Precision, Recall, and mAP to assess the model's performance.

Step 4: Test and Iterate

- Test the model using new images in the Performance tab.
- Add more images and retrain the model if the accuracy is not satisfactory.

Step 5: Publish the Model

- Click on Publish and assign a name for the iteration.
- Record the Prediction URL and API Key from the Prediction Resources tab.

2. Integrate the Model into Power Apps

Step 1: Create a Power Apps Canvas App

- Log in to Power Apps.
- Create a Canvas App from scratch.

Step 2: Connect to the Custom Vision API

- Go to Data > + Add data > Custom Connector.
- Create a new Custom Connector:
  - Enter the Prediction URL and API Key.(Available in your Custom vision project)
  - Configure GET or POST methods for sending data.

Step 3: Add the Upload Image Control

- Navigate to the Insert tab and add an Upload Image control to your screen.
  - Set UploadButton.Text to "Upload Image".
  - Bind UploadedImage.Image to the control for preview.

Step 4: Send Image to the Custom Vision API

Use a formula like the following to send the image to your API:

ClearCollect(Predictions, CustomVision.PredictImage({ image: UploadedImage.Image }))

- Replace CustomVision.PredictImage with the appropriate API method.
- ClearCollect stores the prediction results in a collection called Predictions.

Step 5: Display Predictions

- Add a Gallery control.
- Set the Items property to Predictions.
- Display the prediction labels and confidence scores.

3. Key Power Apps Formulas

ClearCollect

- Purpose: Creates or replaces a collection.
- Example:

ClearCollect(MyCollection, DataSource)

Set

- Purpose: Assigns a value to a global variable.
- Example:

Set(IsImageUploaded, true)

Filter

- Purpose: Filters a data source based on specified conditions.
- Example:

Filter(Predictions, Confidence > 0.8)

Concat

- Purpose: Concatenates values into a single string.
- Example:

Concat(Predictions, TagName & ": " & Confidence, ", ")

4. Tips for Testing and Debugging

Validate API Integration

- Use Postman or similar tools to test the API URL and key before integrating it into Power Apps.

Check Error Messages

- Utilize the Monitor tool in Power Apps to trace API call issues.

Handle Low Confidence Predictions

- Set a confidence threshold (e.g., 80%) to filter out unreliable results.

Debug Image Upload Issues

- Ensure images are in supported formats such as JPG or PNG.

5. Practical Advice

High-Quality Data

- Use varied, high-resolution images during training to improve predictions.
- Include edge cases and diverse backgrounds to avoid overfitting.

Simplify the UI

- Use icons and progress indicators for better user experience.
- Display error messages for invalid inputs or API failures.

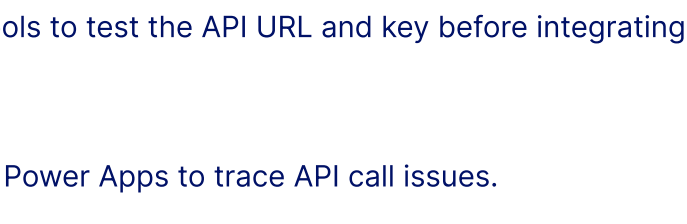
Maintain Performance

- Limit API calls by using the OnSelect property for actions, not OnChange.
- Optimize collections and galleries when dealing with large datasets.

Training with Microsoft Learn

Customize a canvas app in Power Apps

Classify images with Azure AI Custom Vision



Get Microsoft Certified: Power Platform

Microsoft Certified: Power Platform Fundamentals

Microsoft Certified: Power Platform Developer Associate

PL-600: Microsoft Power Platform Solution Architect

Created by: Ayodhya Weerabahu
Microsoft Learn Student Ambassador