Object Detection With Anki's AI Robot

COZMO®

Anne Warren

The Origin of the Idea

Computer vision is a hot and growing field of Artificial Intelligence which complexity and worldwide applicability triggered my interest.

The global computer vision market reached 11.5 billions in 2020.

The applications:

- Object Detection (cats 😉 , cells, tumors, people, guns, people with guns…
- Medical training, robotic surgery,
- Biometric authentication,
- Sport activity tracking,
- Movement analysis,
- Disease progression scoring,
- Self-driving cars,
- Security, video surveillance,
- ….etc…

Cozmo is an AI Robot made by Anki that was first commercialized for Christmas 2016. I bought my first Cozmo in 2017, starring here.

Anki sold 1.5 millions robots and released a Cozmo SDK compatible with Python.

The applications:

- Entertainment for both humans and pets,
- Education by fostering interest for coding, robotics and technologies via the SDK,
- Education by being a visual, interactive, fun support for kids during educational activities via the “Education Mode” and curriculum developed by Cozmo enthusiasts.

https://pisces.bbystatic.com/image2/BestBuy_US/images/products/6006/6006503ld.jpg

The Plan

SAMPLE IMAGES

Cozmo learns to identify an object which holds a sign. The sign is a letter- C, O, Z, M.

KERAS MODEL

The 4 classes of images are labeled and subsequently loaded into a Keras CNN model.

APPLY MODEL

When asked to identify one sign among the four, Cozmo finds the correct sign, says what it is and is happy!

Cozmo SDK

Python scripting is done as usual on the computer using the Cozmo library.

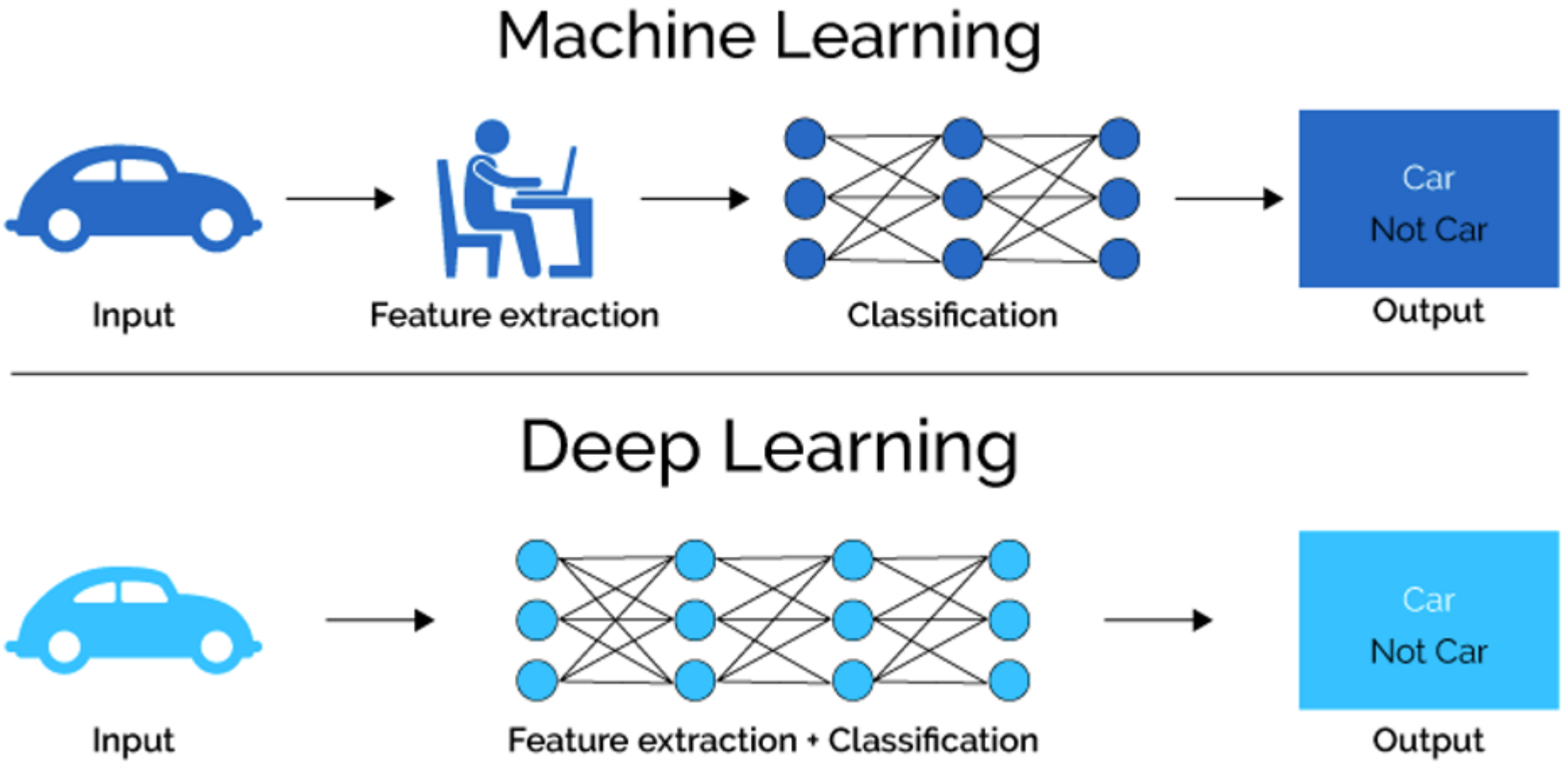
The communication between the computer and Cozmo is done by a cell phone in developer mode.

The interface is managed by the Cozmo application in SDK mode.

A viewer allows to see in real-time what Cozmo sees while it is having fun executing the program. Everything Cozmo does via the entertainment side of the application can be performed through the SDK … and even more!

https://ankicozmorobot.com/wp-content/uploads/2018/01/sdk-setup.jpg

Why Deep Learning?



https://xd.adobe.com/ideas/principles/emerging-technology/what-is-computer-vision-how-does-it-work/

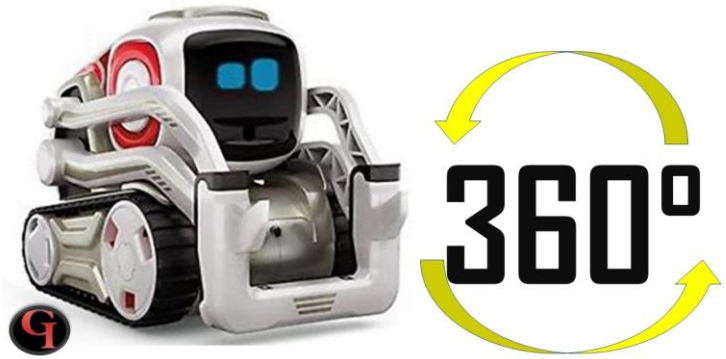
Sampling Strategy

Sampling in circle with Cozmo:

- Cozmo is about 10 cm long,
- Cozmo's centre of rotation is in "his middle",
- Add 5 cm to the distance to be completed.

Sampling in straight line with Cozmo:

- Floor type influences Cozmo's driving,
- Adjust accordingly.



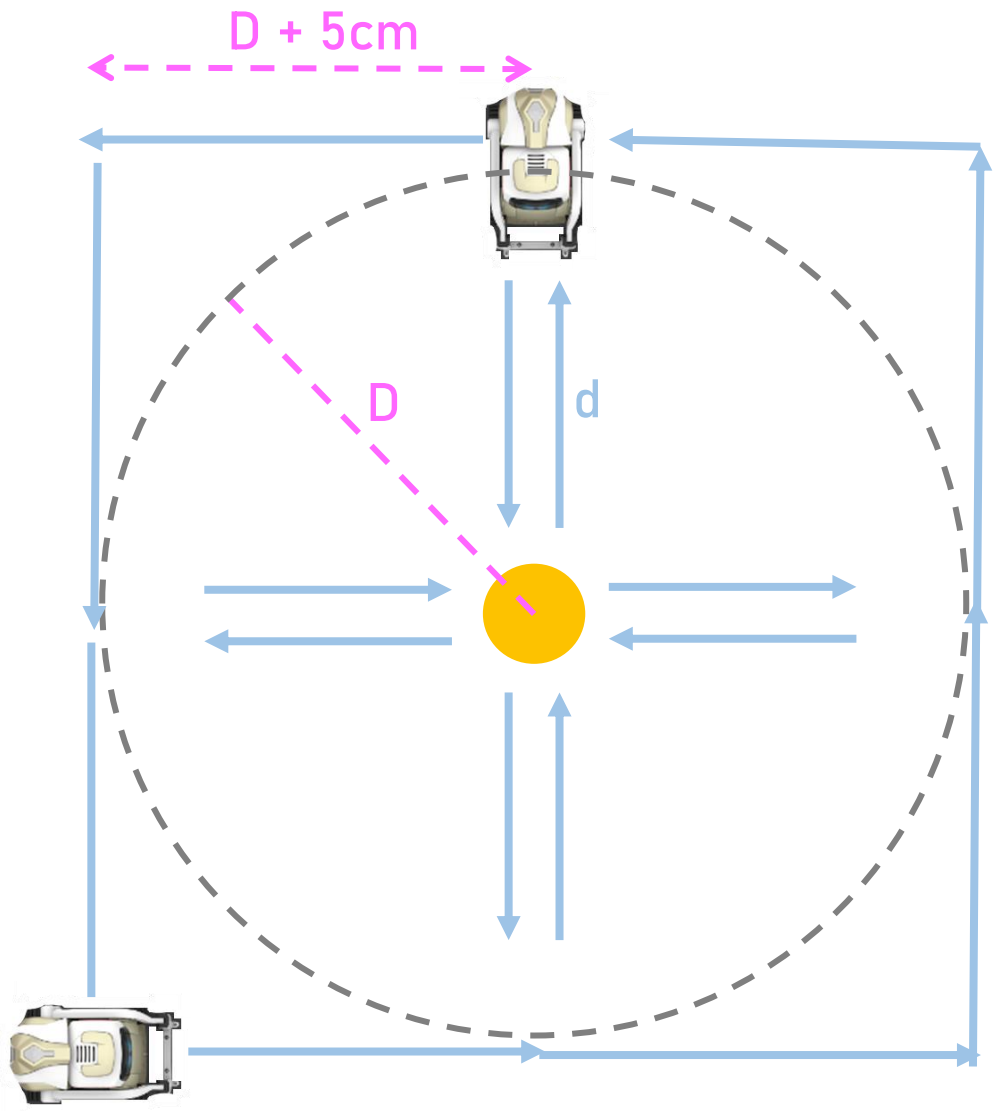
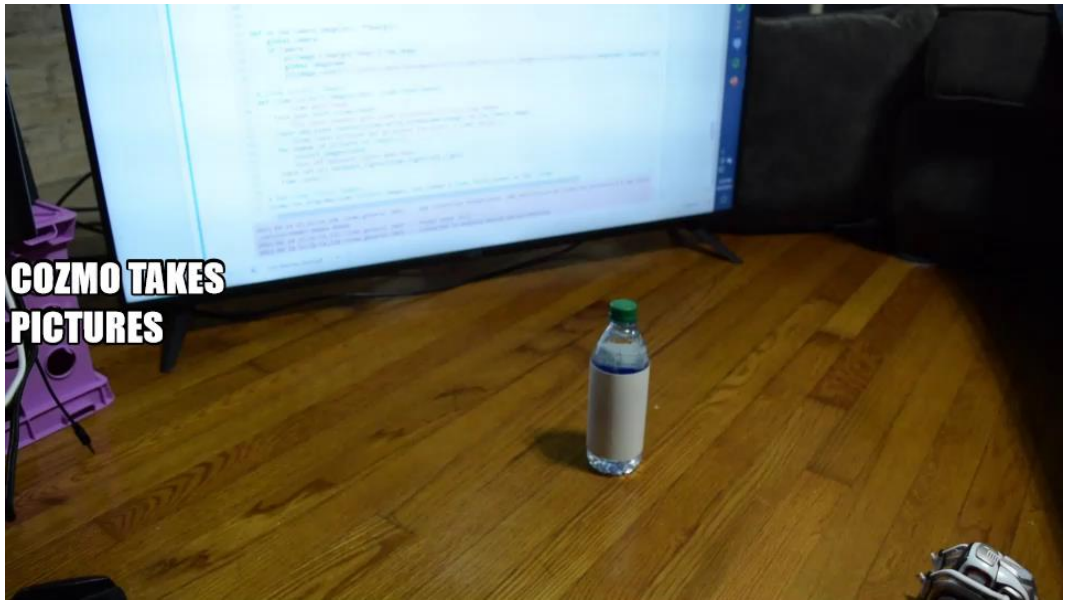


Image from youtube video "Cozmo selfware robot 360 video image can be found here https://i.ytimg.com/vi/_vc4nOx6udU/maxresdefault.jpg

Sampling Coding & Video

```
# Cozmo takes photos
def collect_images(robot: cozmo.robot.Robot):
    ''' Cozmo's backpack lights are blue before taking pictures'''
    robot.set_all_backpack_lights(cozmo.lights.blue_light)
    time.sleep(2)
    ''' Cozmo is placed at 30 cm of the object and takes his first videoshot'''
    ''' first Turn on Cozmo's camera'''
    global camera
    camera = True
    time.sleep(2)
    ''' second Cozmo moves forward 20cm'''
    robot.drive_straight(distance_mm(200), speed_mmps(150), False,False,0).wait_for_completed()
    ''' third Cozmo moves backward to its initial location'''
    robot.drive_straight(distance_mm(-200), speed_mmps(150), False,False,0).wait_for_completed()
    ''' fourth save and Turn off Cozmo's camera'''
    latest_image = robot.world.latest_image.raw_image
    latest_image.convert('L').save(saved_images_path+'image.jpeg')
    camera = False
    time.sleep(2)
    ''' last Cozmo gets into the next position. Left Turns are positive. Right turns are negative.'''
    robot.turn_in_place(degrees(-90)).wait_for_completed()
    robot.drive_straight(distance_mm(350), speed_mmps(150), False,False,0).wait_for_completed()
    robot.turn_in_place(degrees(90)).wait_for_completed()
    robot.drive_straight(distance_mm(350), speed_mmps(150), False,False,0).wait_for_completed()
    robot.turn_in_place(degrees(90)).wait_for_completed()
```



Images

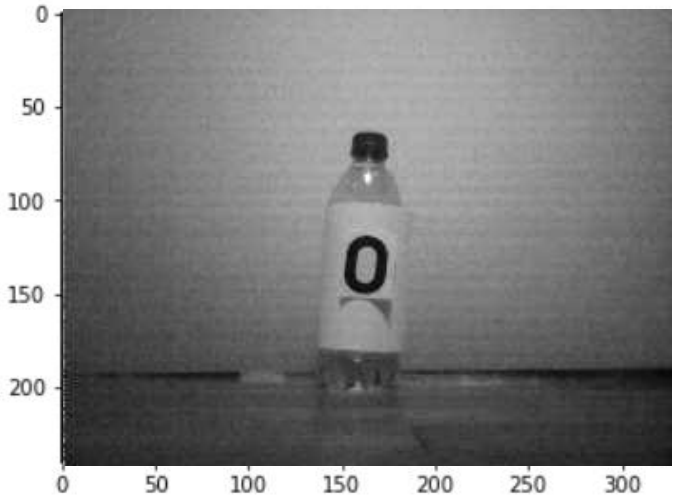
Image attributes: (240, 320, 3)

Key points of sampling and selection:

- Avoid crowded background,
- Promote contrast,
- Remove odd ones.

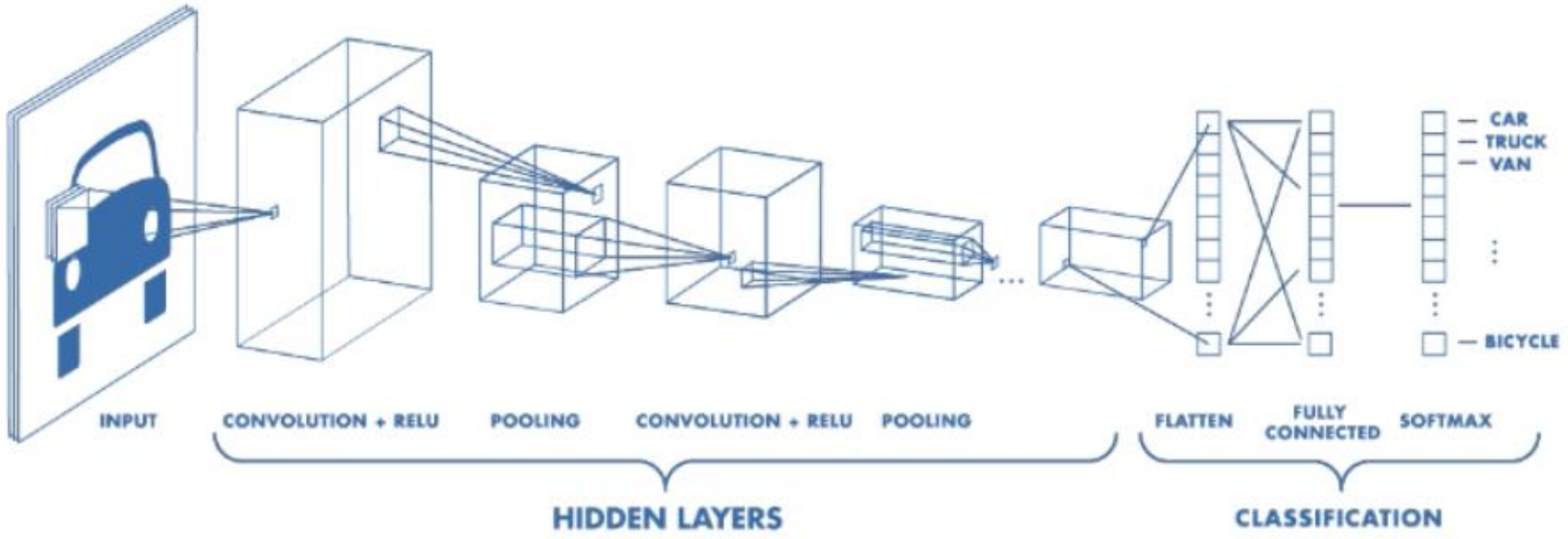
Preparation of images:

- Minimal because it is CNN,
- Keras preprocessing tools.

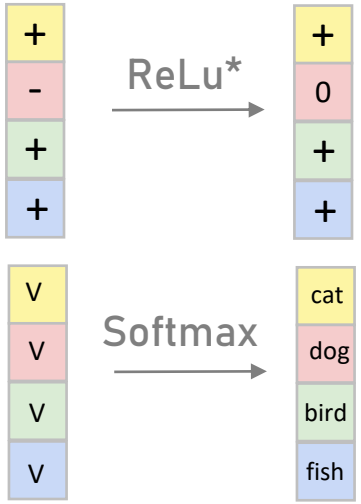


```
train_img = ImageDataGenerator(rescale=1./255,  shear_range=0.2,
                               zoom_range=0.2, horizontal_flip=True)

test_img = ImageDataGenerator(rescale=1./255)

train_generator = train_img.flow_from_directory(
                    path_train_img, target_size=(img_width, img_height),
                    batch_size=batch_size, class_mode='categorical')

validation_generator = test_img.flow_from_directory(
                    path_validation_img, target_size=(img_width, img_height),
                    batch_size=batch_size,  class_mode='categorical')
```

How The CNN Keras Model Works

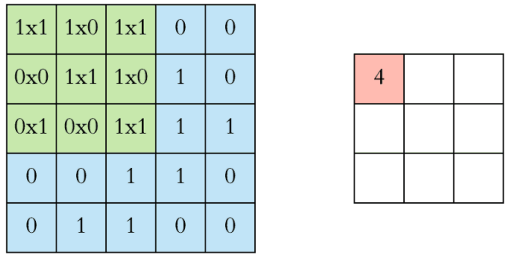


Activations

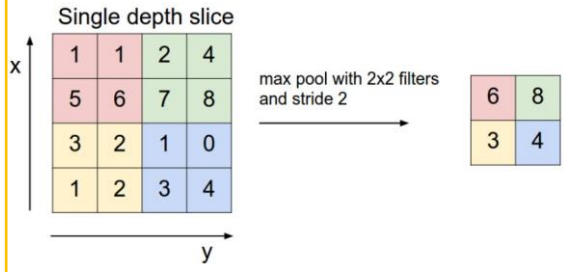


*Rectified Linear Unit

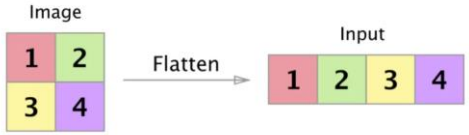
Convolutional Layer



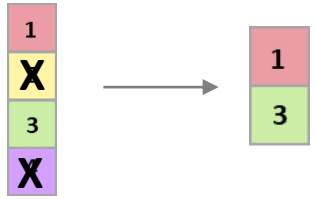
Max Pooling Layer



Flattening Layer



Dropout Layer



Architecture of a CNN https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-1489512765771.html-network
https://www.jessicayung.com/explaining-tensorflow-code-for-a-convolutional-neural-network/ https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/

The Model

```
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(240,320,3)))
model.add(Activation('relu'))
```

1st convolutional layer with 32 filters of size 3x3 with ReLu activation

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

Subsample by taking the max pixel in each 2x2

```
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
```

2nd convolutional layer with 32 filters of size 3x3 with ReLu activation

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

Subsample by taking the max pixel in each 2x2

```
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
```

3rd convolutional layer with 64 filters of size 3x3 With ReLu activation

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

Subsample by taking the max pixel in each 2x2

```
model.add(Flatten())
```

Turn 2D data into 1D to feed subsequent layers

```
model.add(Dense(64))
model.add(Activation('relu'))
```

Fully connected layer with 64 filters

```
model.add(Dropout(0.5))
```

50% of the neurons get to contribute

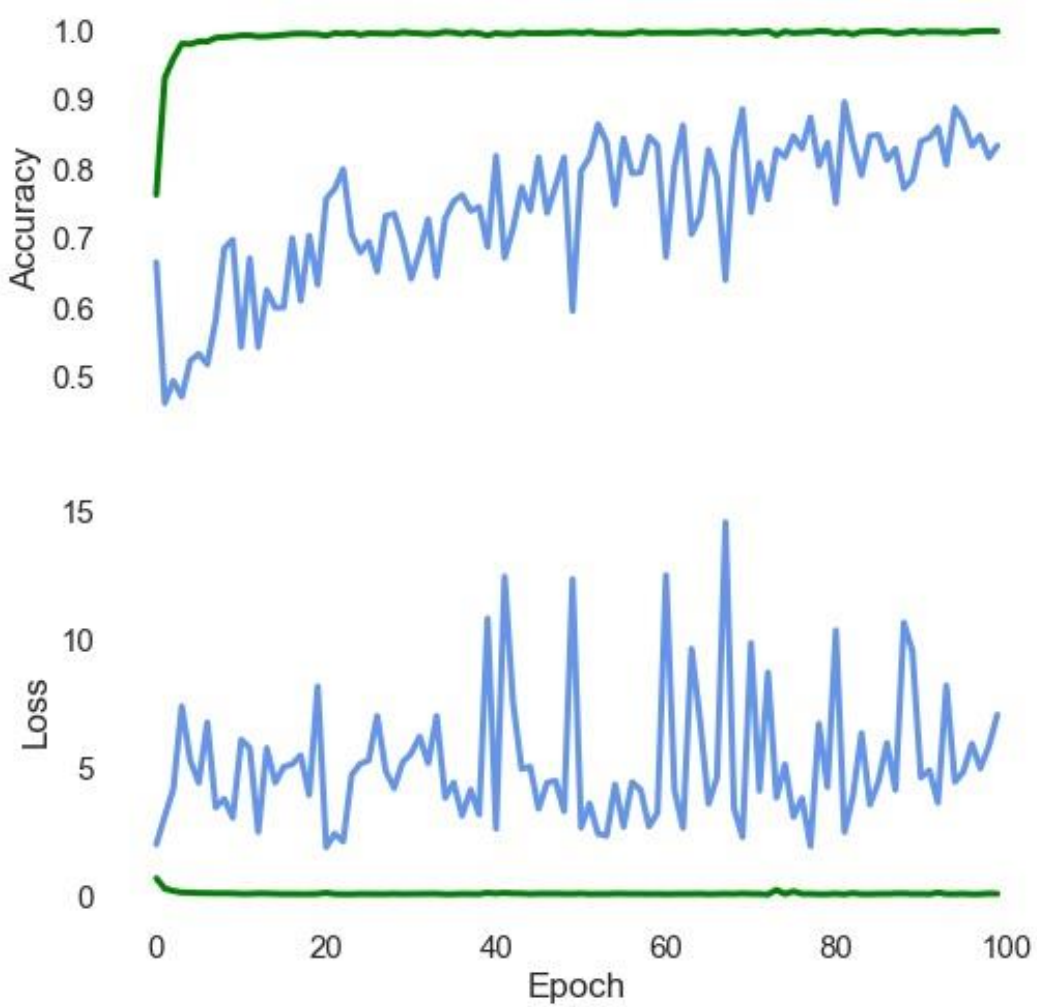
```
model.add(Dense(4))
model.add(Activation('softmax'))
```

Output one out of four classes with the Softmax activation.

Model Performance and Metrics

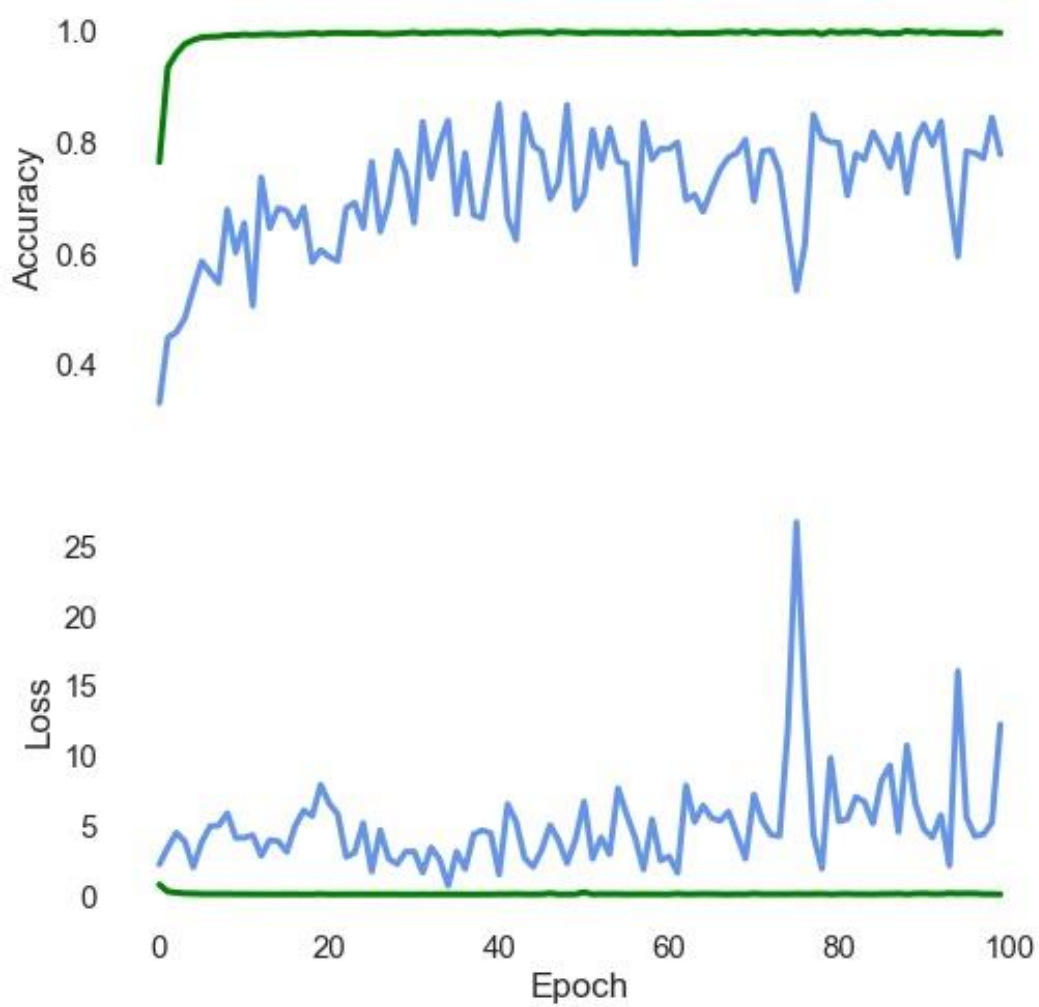
Without Callbacks Accuracy = 83%

loss: categorical_crossentropy , optimizer: rmsprop
4,646 training images and 2,795 validation images



With Callbacks Accuracy = 77%

loss: categorical_crossentropy , optimizer: rmsprop
4,646 training images and 2,795 validation images

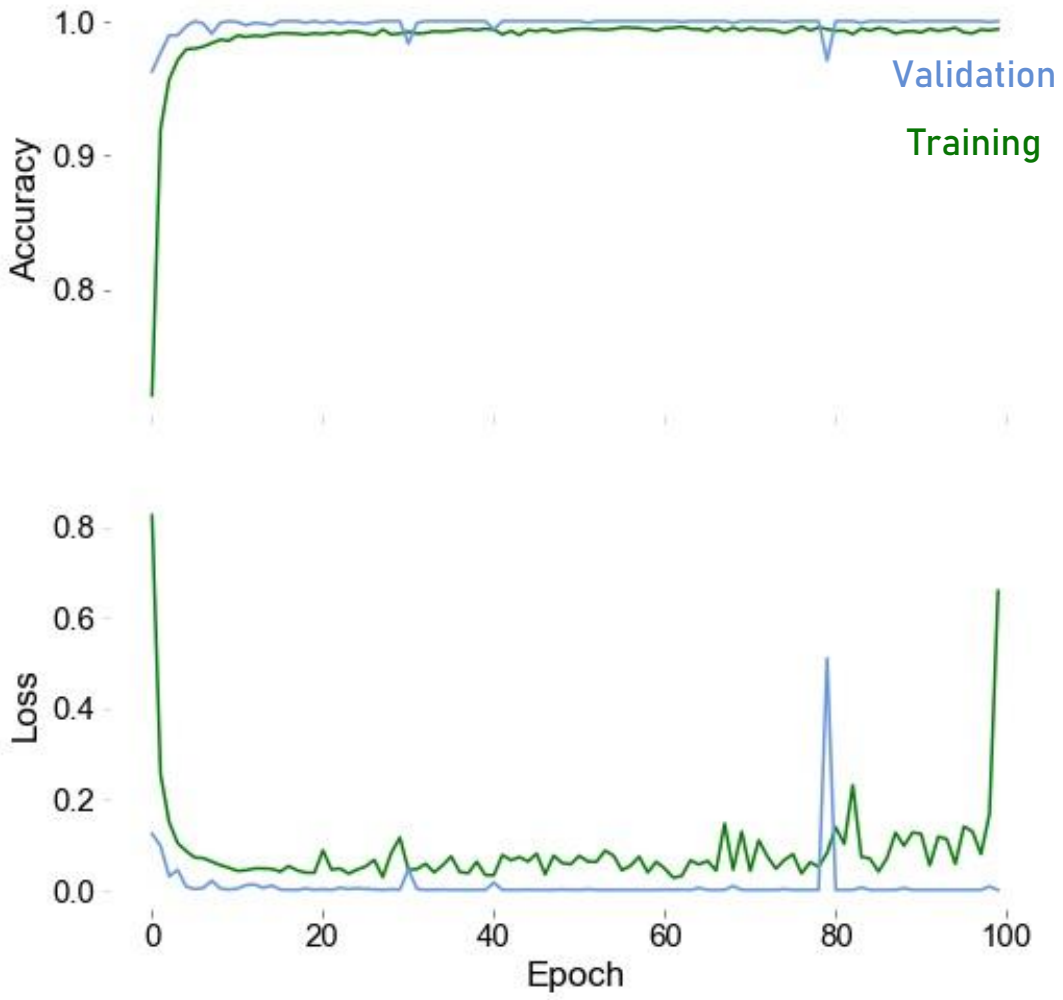


Model Performance and Metrics

After Shuffle Accuracy = 99.8%

loss: categorical_crossentropy , optimizer: rmsprop

7,352 training images and 1,600 validation images



Shuffling the image deck made a huge difference. The accuracy moved up to 99.8%. The validation curve (blue) shows higher accuracy than the training curve (green) because the model has a dropout layer (50% drop).

The model reaches optimum predictive capability after 10 epochs. Interestingly both curves show that the model is running into issues around 30 and 80 epochs.

Model Application

Cozmo looks at an item and collects new images.

The images are normalized and sent to the model.

The model classifies the images and provides an answer.

Cozmo says what it is.

Conclusion And Future Work

Simple Convolutional Neural Networks are relatively easy to build and make very powerful models for image classification and object detection.

The sequential model used in this project has good accuracy (>80%) but the accuracy curve shows that there is a need to shuffle the images between training and validation. Adding more images would be also beneficial. After shuffling the model reaches 99.8%.

As of now Cozmo just says the prediction of the model or “I don’t know this one” if the class is not known from the model yet. A confidence level using the model uncertainty and data uncertainty could be added to the program to prevent Cozmo from saying a wrong prediction.

New features will be added to the program:

- new classes (the whole alphabet),
- Cozmo will read words by putting together the letters it sees,
- Cozmo will wander around and be triggered by an object/letter it recognizes.