

SUPPORTIVE TOOLING FOR ML

Traditional Python

- No type information for x
- Extensive type comments
- Tedious
- Error Prone

```

41 # MNIST data input is a 1-D vector of 784 features (28*28 pixels)
42 # Reshape to match picture format [Height x Width x Channel]
43 # Tensor input become 4-D: [Batch Size, Height, Width, Channel]
44 z = tf.reshape(x, shape=[-1, 28, 28, 1])
45
46 # Convolution Layer with 32 filters and a kernel size of 5
47 conv1 = tf.layers.conv2d(z, 32, 5, activation=tf.nn.relu)
48 # Max Pooling (down-sampling) with strides of 2 and kernel size of 2
49 conv1 = tf.layers.max_pooling2d(conv1, 2, 2)
50
  
```

Python with Ariadne

- Type information exposed
- Type comments unnecessary
- Effortless
- Automatically Maintained

```

41 # MNIST data input is a 1-D vector of 784 features (28*28 pixels)
42 # Reshape to match picture format [Height x Width x Channel]
43 x type: pixel[n][28 * 28]   come 4-D: [Batch Size, Height, Width, Channel]
44 z = tf.reshape(x, shape=[-1, 28, 28, 1])
45
46 # Convolution Layer with 32 filters and a kernel size of 5
47 conv1 = tf.layers.conv2d(z, 32, 5, activation=tf.nn.relu)
48 # Max Pooling (down-sampling) with strides of 2 and kernel size of 2
49 conv1 = tf.layers.max_pooling2d(conv1, 2, 2)
50
  
```

IMPLEMENTATION



WALA
T. J. WATSON LIBRARIES FOR ANALYSIS

- full-featured, state-of-the-art analysis framework
- used for JavaScript in tools such as AppScan
- extend to analyze Python, using flexible CAst front end



Language Server Protocol

- Communication between IDEs and Information Providers
- Use JSON-RPC to decouple IDEs from Analysis
- Originated in VSCode, now increasingly supported

Currently Supported



Atom



VSCode



PyCharm



Sublime



Emacs



Vim



NeoVim

Coming Soon



Jupyter



Spyder



Eclipse

AVAILABLE FEATURES

Backward Slices

- represent the flow of data
- track tensors flowing through the program
- in this case, globally through object 'x_dict'

```

• buggy_convolutional_network.py(37, 13): x_dict['images']
• buggy_convolutional_network.py(33, 14): x_dict
• buggy_convolutional_network.py(78, 28): features
• buggy_convolutional_network.py(73, 14): features
• buggy_convolutional_network.py(117, 7): {'images': mnist.train.images}
• buggy_convolutional_network.py(117, 18): mnist
• buggy_convolutional_network.py(117, 18): mnist.train.images
  
```

- Uses WALA slicer
- Whole program
- Flow & context sensitive

Code Lenses

- detail assignments
- expose the shape of assigned tensor shown
- describe data throughout the program

```

z: pixel[?][28][28][1]
z = tf.reshape(x, shape=[-1, 28, 28, 1])

conv1: pixel[?][28][28][1]
# Convolution Layer with 32 filters and a kernel size of 5
conv1 = tf.layers.conv2d(z, 32, 5, activation=tf.nn.relu)
  
```

- Uses WALA dataflow
- Propagates tensor shapes
- Usual fixpoint dataflow
- Models tensor operations

Hover Tips

- show tensor variables
- display when users hover over code
- provide detail only when requested

```

reshape(x, shape=[ z
                    type: pixel[?][28][28][1]
                    target: tensorflow.functions.conv2d
                    tf.layers.conv2d(z, 32, 5, activation=tf.nn.re
  
```

Auto Fixes

- can provide fixes

```

bad_conv1 = tf.layers.conv2d(x, 32, 5, activation=tf.nn.relu)
tf.reshape(x, [-1, 28, 28, 1])
  
```

- Tensor models for misuse

Error Reports

- are based on analysis of tensor errors
- integrate with IDE error reporting

```

buggy_convolutional_network.py ~/git/ML/com.ibm.wala.cast.python.test/data 4
[Ariadne] Cannot reshape pixel[n][28 * 28] to pixel[?][11][28][1] (38, 28)
[Ariadne] Bad type to convolve pixel[n][28 * 28], needs 4 dimensions (possible fix: tf.reshape(x, [-1, 28, 28... (55, 38)
  
```

- Heuristics to suggest fixes