Image Classification

Image Classification

Given an image, identifying which object is in the image

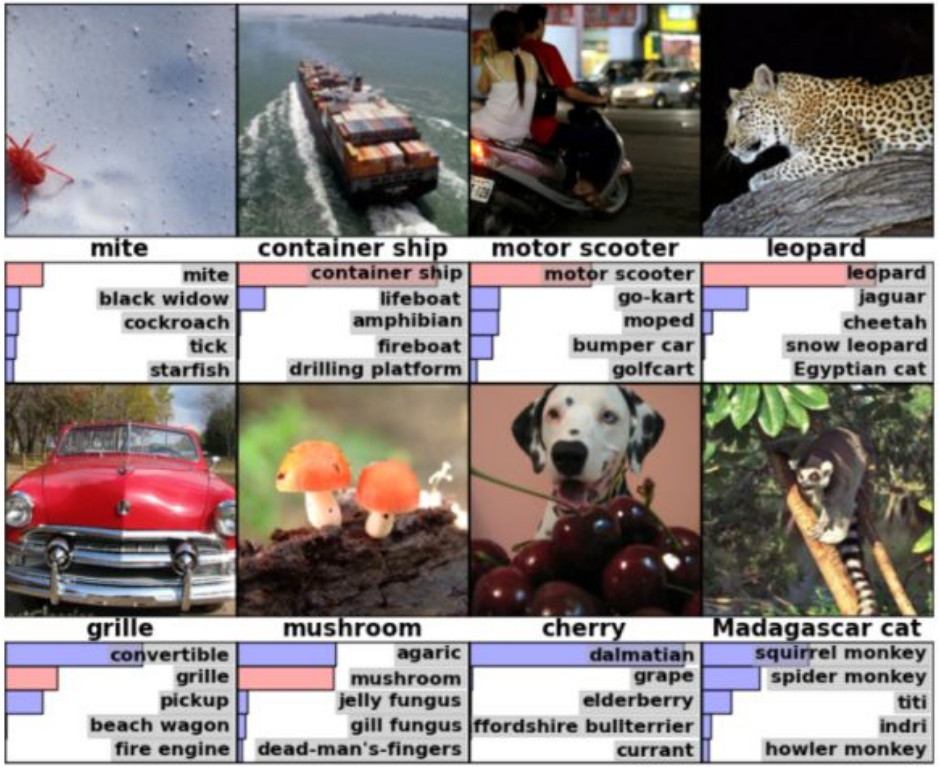
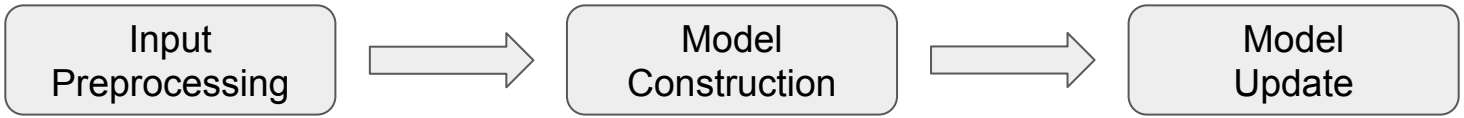


Image Classification



Data loader

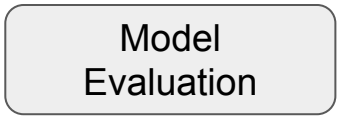
- Load batch data
- Preprocess the data
  - normalization
  - random crop
  - flipping

Forward path

- Build input placeholders
- Build network
  - convolution
  - linear (fc)
  - pooling
  - dropout
  - activation function
- Build loss
  - cross-entropy

Backward path

- Compute gradient
  - gradient clip
- Update model
  - SGD, Adam, Adagrad, etc
  - Weight decay



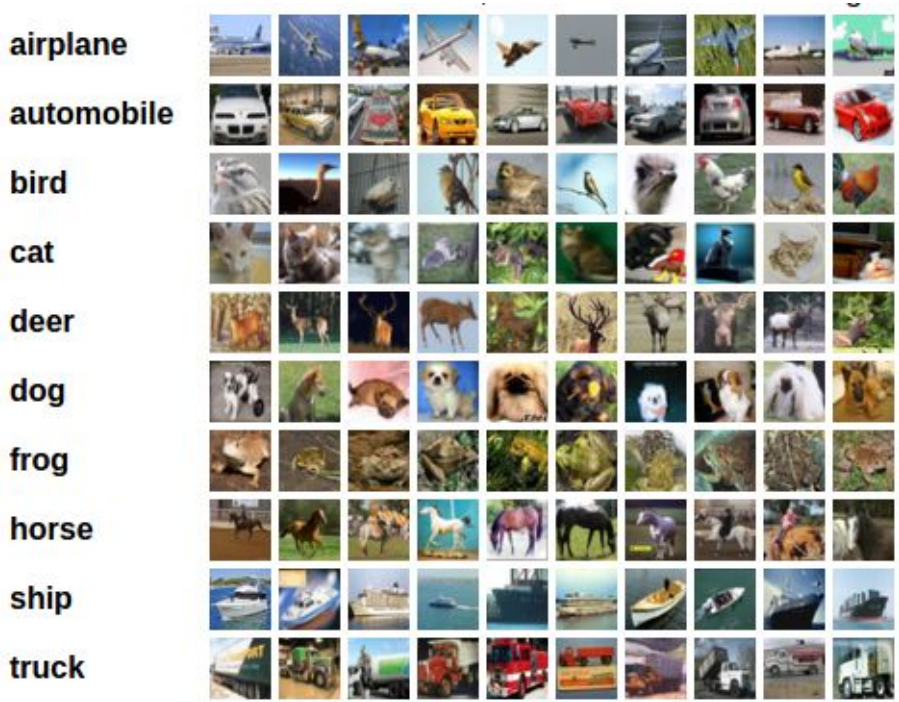
Measure the accuracy

- # of correct examples / # of total examples

Input Preprocessing

CIFAR-10 Dataset

- **10 classes** - airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck
- **32x32 color images** with 6,000 images per class (50,000 training / 10,000 test)



Input Preprocessing

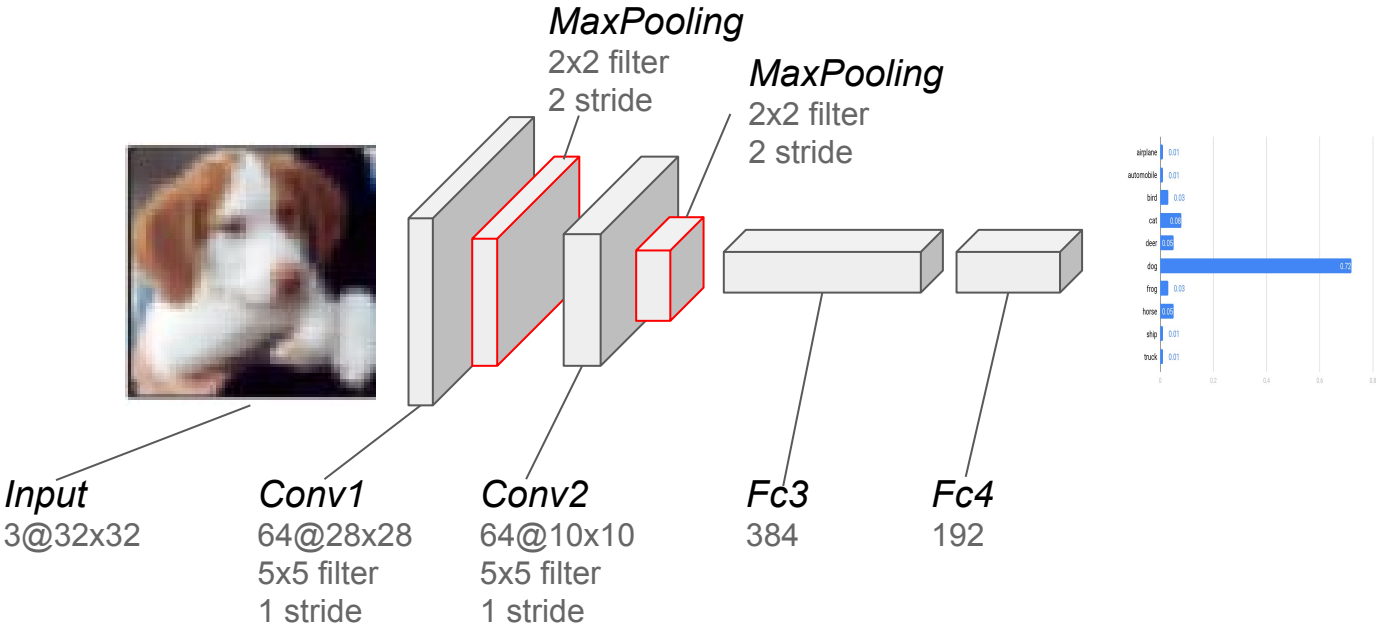
Data loader (cifar10_loader.py) does

- Reshape a given vector of 3,072 size into a tensor of 3x32x32 size
  - (channel, width, height)
- Transpose (channel, width, height) into (width, height, channel)
- Load data of batch size using get_batch()

Download and check the dataset in the ipython notebook
(002_image_classification/image_classification.ipynb)

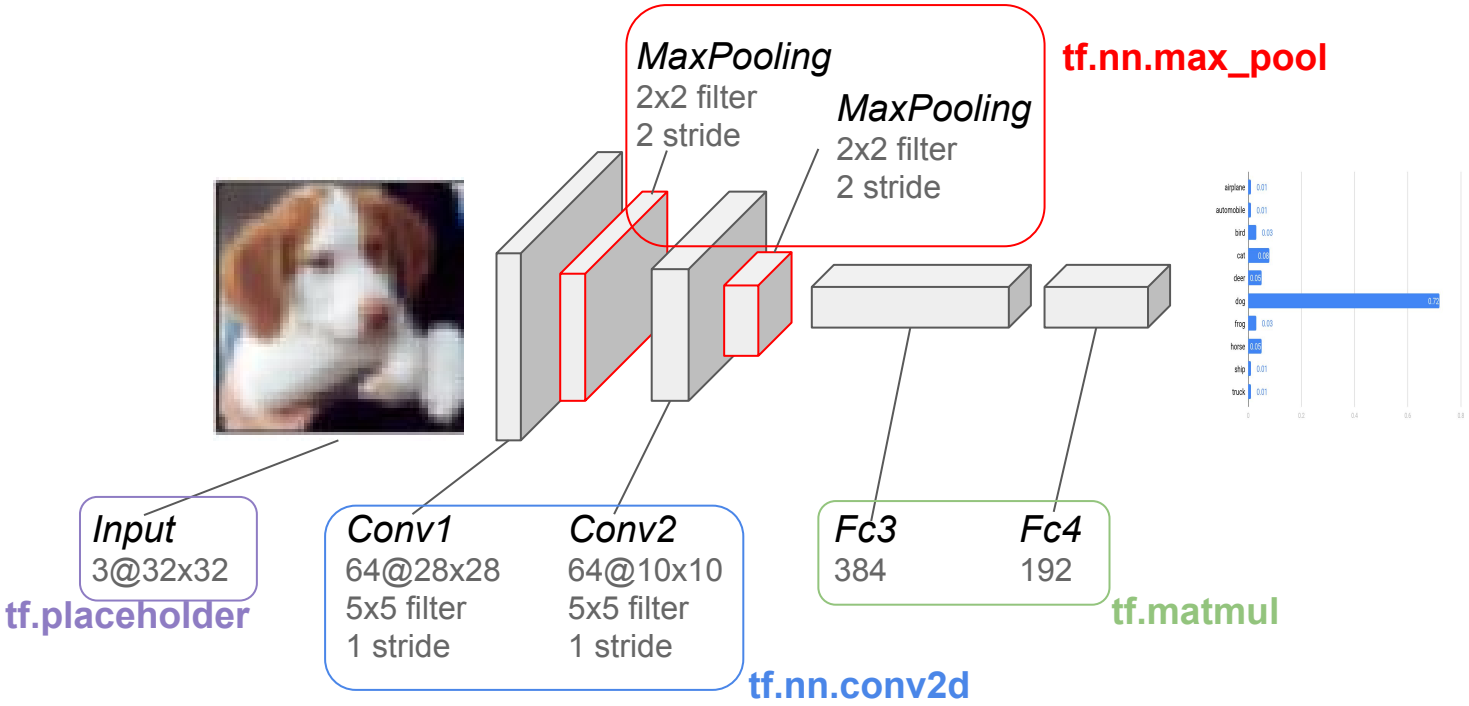
Model Construction

2 convolutional layers + 2 fully connected layers + classification layer



Model Construction

How to implement the layers in TensorFlow?



Model Construction

How to implement the layers?

Convolutional Layer - tf.nn.conv2d()

```
# Create variables
filter = tf.get_variable("weights", [filer_size,filter_size,
inp_dim, out_dim])
bias = tf.get_variable("biases", [out_dim])

# Compute convolution
conv = tf.nn.conv2d(input, filter)
conv = tf.nn.bias_add(conv, bias)
```

Fully Connected Layer - tf.matmul()

```
# Create variables
weight = tf.get_variable("weights", [inp_dim, out_dim])
bias = tf.get_variable("biases", [out_dim])

# Compute convolution
fc = tf.matmul(input, weight)
fc = tf.nn.bias_add(fc, bias)
```

If you want to apply activation function (relu, tanh)?

```
conv = tf.nn.relu(conv)
conv = tf.tanh(conv)
```

```
fc = tf.nn.relu(fc)
fc = tf.tanh(fc)
```

Max pooling?

```
# kernel size 2 and stride 2
pool = tf.nn.max_pool(conv, ksize=[1,2,2,1],
strides=[1,2,2,1])
```

Dropout?

```
drop_fc = tf.nn.dropout(fc, keep_prob)
```

Model Construction

We should always define variables manually? **No**

There are three libraries based on TensorFlow which are **simple and concise**!

```
# Create variables
filter = tf.get_variable("weights", [filer_size,filter_size,
inp_dim, out_dim])
bias = tf.get_variable("biases", [out_dim])

# Compute convolution
conv = tf.nn.conv2d(input, filter)
conv = tf.nn.bias_add(conv, bias)

# Apply activation function
conv = tf.nn.relu(conv)
conv = tf.tanh(conv)

# Apply dropout
drop_conv = tf.nn.dropout(conv, keep_prob)
```

wrapper function

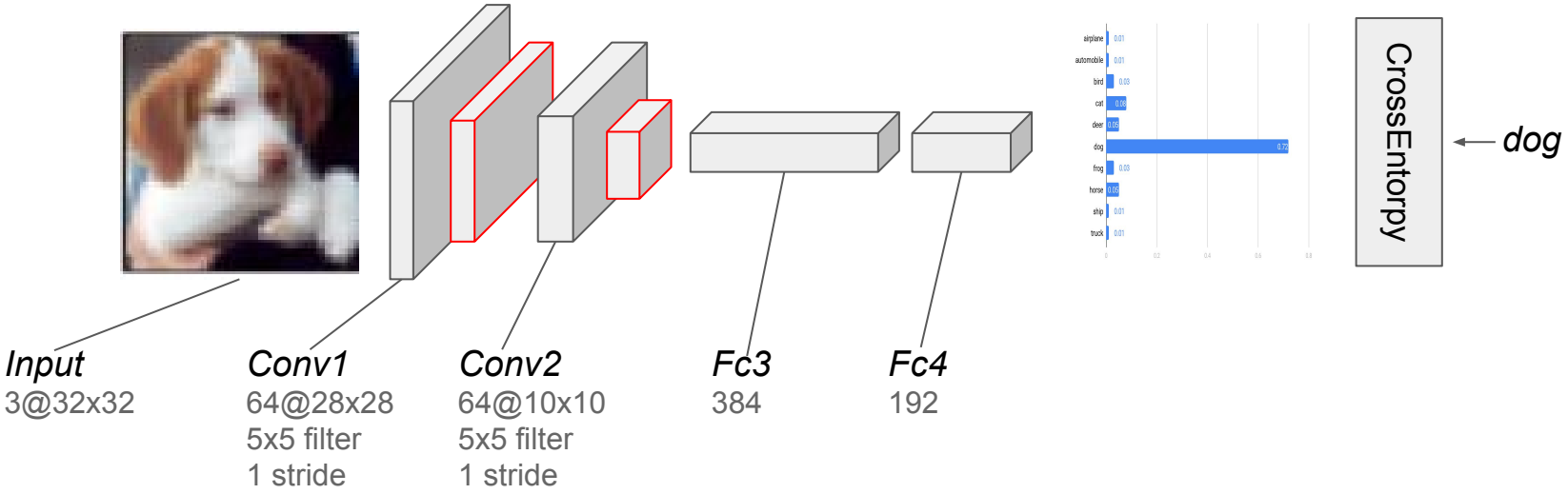
1. TFLearn
   - tflearn.layers.conv.conv_2d()
2. Keras
   - tf.contrib.keras.layers.Conv2D()
3. TF-Slim
   - tf.contrib.slim.conv2d()

Note that we do not cover above libraries in this course

Model Update

Define loss function (cross-entropy loss)

Define optimizer (Adam)



Model Update

How to define loss function and optimizer in TensorFlow?

Loss Function

```
# Compute cross-entropy loss
cross_entropy =
tf.nn.sparse_softmax_cross_entropy_with_logits(
labels, logits)
loss = tf.reduce_mean(cross_entropy)
```

* cross-entropy loss?
  - LogSoftMax + ClassNLLCriterion
  - loss(x, class)
    = -log(exp(x[class]) /∑exp(x[j]))
    = -x[class] + log($\sum_j$ exp(x[j]))

Optimizer

```
# Create optimizer
optimizer = tf.train.AdamOptimizer(learning_rate)

# Two methods exists
# 1. compute gradients and update model
grads_vars = optimizer.compute_gradients(loss)
train_op = optimizer.apply_gradients(grads_vars)

# 2. Update model with one function
train_op = optimizer.minimize(loss)
```

* other optimizers?
  - tf.train.GradientDescentOptimizer()
  - tf.train.RMSPropOptimizer()
  - tf.train.AdagradOptimizer()

Now, we can train model with following line

```
sess = tf.Session()
sess.run(fetches=train_op, feed_dict=inputs)
```

Misc

How to show the variables defined in the network (graph)?

```
for var in tf.global_variables():
    print(var)
```

* global_variables() returns the variables that are created by tf.Variable() or tf.get_variable()

* We can show operations using tf.get_default_graph().get_operations()

```
Tensor("conv1/weights/read:0", shape=(5, 5, 3, 64), dtype=float32)
Tensor("conv1/biases/read:0", shape=(64,), dtype=float32)
Tensor("conv2/weights/read:0", shape=(5, 5, 64, 64), dtype=float32)
Tensor("conv2/biases/read:0", shape=(64,), dtype=float32)
Tensor("fc3/weights/read:0", shape=(1600, 384), dtype=float32)
Tensor("fc3/biases/read:0", shape=(384,), dtype=float32)
Tensor("fc4/weights/read:0", shape=(384, 192), dtype=float32)
Tensor("fc4/biases/read:0", shape=(192,), dtype=float32)
Tensor("fc5/weights/read:0", shape=(192, 10), dtype=float32)
Tensor("fc5/biases/read:0", shape=(10,), dtype=float32)
```

How to access weight parameters?

```
sess = tf.Session()                                          # create session
g = tf.get_default_graph()                                   # get the current graph model
w_tensor = g.get_tensor_by_name("fc5/weights/read:0")        # obtain a tensor variable using the name
w = w_tensor.eval(session=sess)                              # get the tensor by running session
print(w.shape)
print(w[0, :10])
```

```
(192, 10)
[ 0.00417436  0.01035099  0.00198316  0.00643856  0.00888029  0.00903297
 -0.0122347  -0.01635222 -0.01234471  0.00342033]
[ 0.00417436  0.01035099  0.00198316  0.00643856  0.00888029  0.00903297
 -0.0122347  -0.01635222 -0.01234471  0.00342033]
```

Check the code image_classification.ipynb

Train the model and evaluate it

Exercises

1. Use residual block instead of simple convolutional layer
   a. https://arxiv.org/pdf/1512.03385.pdf
   b. https://arxiv.org/pdf/1603.05027.pdf