Deep Learning with TensorFlow

Deep Learning — Units 7 & 8

Dr. Jon Krohn
`jon@untapt.com`

Slides available at `jonkrohn.com/talks`

March 24th, 2018

NYC DATA SCIENCE ACADEMY

Outline

NYC DATA SCIENCE ACADEMY

Outline

NYC DATA SCIENCE ACADEMY

Outline

NYC DATA SCIENCE ACADEMY

Outline

Outline

NYC DATA SCIENCE ACADEMY

Outline

Outline

NYC DATA SCIENCE ACADEMY

Outline

Outline

NYC DATA SCIENCE ACADEMY

Assessing

Your Deep Learning Project III

Assessing

Your Deep Learning Project III

Where are you at with respect to the following?

1. Splitting your data
   - training set (80% — for optimizing parameters)
   - validation set (10% — for hyperparameters)
   - test set (10% — don't touch yet!)
2. Building and assessing architecture
   - get above chance (simplifying problem, if necessary)
   - do existing performance benchmarks exist?
   - if not, use a simple architecture as benchmark

Assessing

Your Deep Learning Project III

Where are you at with respect to the following?

1. Splitting your data
   - training set (80% — for optimizing parameters)
   - validation set (10% — for hyperparameters)
   - test set (10% — don't touch yet!)
2. Building and assessing architecture
   - get above chance (simplifying problem, if necessary)
   - do existing performance benchmarks exist?
   - if not, use a simple architecture as benchmark

Assessing

Your Deep Learning Project III

Where are you at with respect to the following?

1. Splitting your data
   - training set (80% — for optimizing parameters)
   - validation set (10% — for hyperparameters)
   - test set (10% — don't touch yet!)
2. Building and assessing architecture
   - get above chance (simplifying problem, if necessary)
   - do existing performance benchmarks exist?
   - if not, use a simple architecture as benchmark

Assessing

Your Deep Learning Project III

Where are you at with respect to the following?

1. Splitting your data
   - training set (80% — for optimizing parameters)
   - validation set (10% — for hyperparameters)
   - test set (10% — don't touch yet!)
2. Building and assessing architecture
   - get above chance (simplifying problem, if necessary)
   - do existing performance benchmarks exist?
   - if not, use a simple architecture as benchmark

Assessing

Your Deep Learning Project III

Where are you at with respect to the following?

1. Splitting your data
   - training set (80% — for optimizing parameters)
   - validation set (10% — for hyperparameters)
   - test set (10% — don't touch yet!)
2. Building and assessing architecture
   - get above chance (simplifying problem, if necessary)
   - do existing performance benchmarks exist?
   - if not, use a simple architecture as benchmark

Assessing

Your Deep Learning Project III

Where are you at with respect to the following?

1. Splitting your data
   - training set (80% — for optimizing parameters)
   - validation set (10% — for hyperparameters)
   - test set (10% — don't touch yet!)
2. Building and assessing architecture
   - get above chance (simplifying problem, if necessary)
   - do existing performance benchmarks exist?
   - if not, use a simple architecture as benchmark

Assessing

Your Deep Learning Project III

Where are you at with respect to the following?

1. Splitting your data
   - training set (80% — for optimizing parameters)
   - validation set (10% — for hyperparameters)
   - test set (10% — don't touch yet!)
2. Building and assessing architecture
   - get above chance (simplifying problem, if necessary)
   - do existing performance benchmarks exist?
   - if not, use a simple architecture as benchmark

Assessing

Your Deep Learning Project III

Where are you at with respect to the following?

1. Splitting your data
   - training set (80% — for optimizing parameters)
   - validation set (10% — for hyperparameters)
   - test set (10% — don't touch yet!)
2. Building and assessing architecture
   - get above chance (simplifying problem, if necessary)
   - do existing performance benchmarks exist?
   - if not, use a simple architecture as benchmark

Outline

Leading DL Libraries

	Caffe	Torch	MXNet	TensorFlow
Language	Python, Matlab	Lua, C	Python, R, C++ Julia, Matlab JavaScript, Go Scala, Perl	Python, R, C++ C, Java, Go
Programming Style	Symbolic	Imperative	Imperative	Symbolic
Parallel GPUs: Data	Yes	Yes	Yes	Yes
Parallel GPUs: Model		Yes	Yes	Yes
Pre-Trained Models	Model Zoo	ModelZoo	Model Zoo	github.com/tensorflow/models
For RNNs				Best
High-Level APIs		PyTorch	in-built	Keras, TFLearn

Outline

Outline

NYC DATA SCIENCE ACADEMY

TensorFlow Graphs

1. build graph
2. initialize session
3. fetch and feed data

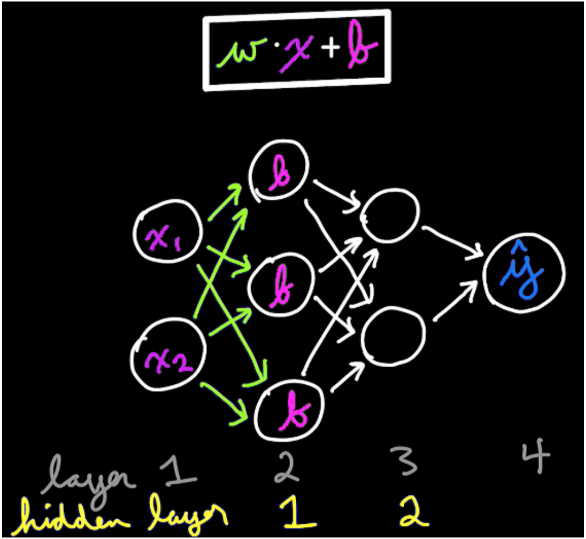
TensorFlow Graphs

1. build graph
2. initialize session
3. fetch and feed data

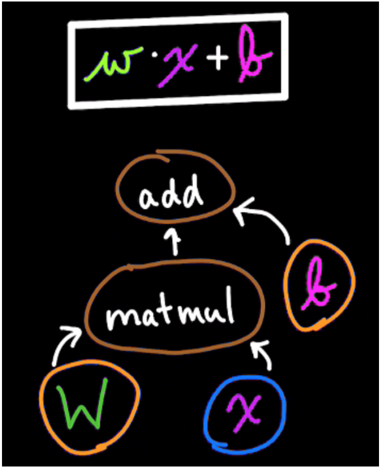
TensorFlow Graphs

1. build graph
2. initialize session
3. fetch and feed data

A Familiar Equation



TensorFlow Graphs



Outline

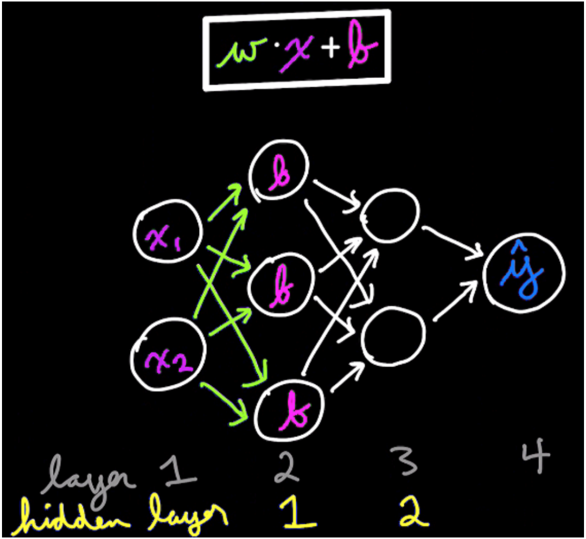
TensorFlow Graph
Programming

[*first TensorFlow graphs* notebook]

Outline

NYC DATA SCIENCE ACADEMY

A Familiar Equation



Neurons in TensorFlow

Programming

[*first TensorFlow neurons* notebook]

Outline

NYC DATA SCIENCE ACADEMY

Outline

Fitting Eight Points

[*point by point intro to TensorFlow* notebook]

Fitting Eight Points

with Tensors

[*tensor-fied intro to TensorFlow* notebook]

Outline

NYC DATA SCIENCE ACADEMY

Fitting Eight Million Points

[*intro to TensorFlow times a million* notebook]

Outline

NYC DATA SCIENCE ACADEMY

Dense Nets

[*intermediate net in TensorFlow* notebook]

Dense Nets

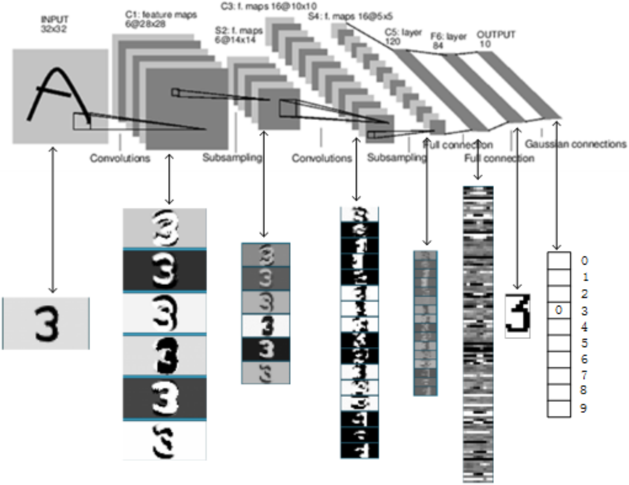
[*deep net in TensorFlow* notebook]

Outline

NYC DATA SCIENCE ACADEMY

LeNet-5

LeCun et al. (1998)



LeNet-5

LeCun et al. (1998)

[*LeNet in TensorFlow* notebook]

Outline

NYC DATA SCIENCE ACADEMY

Outline

NYC DATA SCIENCE ACADEMY

Improving

Your Deep Learning Project IV

Improving
Your Deep Learning Project IV

1. Splitting your data
   - training set (80% — for optimizing parameters)
   - validation set (10% — for hyperparameters)
   - test set (10% — don't touch yet!)
2. Building and assessing architecture
   - get above chance (simplifying problem, if necessary)
   - do existing performance benchmarks exist?
   - if not, use a simple architecture as benchmark
3. Improving performance & tuning hyperparameters in ten steps...

Improving

Your Deep Learning Project IV

1. Splitting your data
    - training set (80% — for optimizing parameters)
    - validation set (10% — for hyperparameters)
    - test set (10% — don't touch yet!)
2. Building and assessing architecture
    - get above chance (simplifying problem, if necessary)
    - do existing performance benchmarks exist?
    - if not, use a simple architecture as benchmark
3. Improving performance & tuning hyperparameters in ten steps...

Improving

Your Deep Learning Project IV

1. Splitting your data
   - training set (80% — for optimizing parameters)
   - validation set (10% — for hyperparameters)
   - test set (10% — don't touch yet!)
2. Building and assessing architecture
   - get above chance (simplifying problem, if necessary)
   - do existing performance benchmarks exist?
   - if not, use a simple architecture as benchmark
3. Improving performance & tuning hyperparameters in ten steps...

Improving

Your Deep Learning Project IV

1. Splitting your data
   - training set (80% — for optimizing parameters)
   - validation set (10% — for hyperparameters)
   - test set (10% — don't touch yet!)
2. Building and assessing architecture
   - get above chance (simplifying problem, if necessary)
   - do existing performance benchmarks exist?
   - if not, use a simple architecture as benchmark
3. Improving performance & tuning hyperparameters in ten steps...

Improving

Your Deep Learning Project IV

1. Splitting your data
   - training set (80% — for optimizing parameters)
   - validation set (10% — for hyperparameters)
   - test set (10% — don't touch yet!)
2. Building and assessing architecture
   - get above chance (simplifying problem, if necessary)
   - do existing performance benchmarks exist?
   - if not, use a simple architecture as benchmark
3. Improving performance & tuning hyperparameters in ten steps...

Improving

Your Deep Learning Project IV

1. Splitting your data
   - training set (80% — for optimizing parameters)
   - validation set (10% — for hyperparameters)
   - test set (10% — don't touch yet!)
2. Building and assessing architecture
   - get above chance (simplifying problem, if necessary)
   - do existing performance benchmarks exist?
   - if not, use a simple architecture as benchmark
3. Improving performance & tuning hyperparameters in ten steps...

Improving

Your Deep Learning Project IV

1. Splitting your data
   - training set (80% — for optimizing parameters)
   - validation set (10% — for hyperparameters)
   - test set (10% — don't touch yet!)
2. Building and assessing architecture
   - get above chance (simplifying problem, if necessary)
   - do existing performance benchmarks exist?
   - if not, use a simple architecture as benchmark
3. Improving performance & tuning hyperparameters in ten steps...

NYC DATA SCIENCE ACADEMY

Improving

Your Deep Learning Project IV

1. Splitting your data
   - training set (80% — for optimizing parameters)
   - validation set (10% — for hyperparameters)
   - test set (10% — don't touch yet!)
2. Building and assessing architecture
   - get above chance (simplifying problem, if necessary)
   - do existing performance benchmarks exist?
   - if not, use a simple architecture as benchmark
3. Improving performance & tuning hyperparameters in ten steps...

Improving

Your Deep Learning Project IV

1. Splitting your data
   - training set (80% — for optimizing parameters)
   - validation set (10% — for hyperparameters)
   - test set (10% — don't touch yet!)
2. Building and assessing architecture
   - get above chance (simplifying problem, if necessary)
   - do existing performance benchmarks exist?
   - if not, use a simple architecture as benchmark
3. Improving performance & tuning hyperparameters in ten steps...

Outline

NYC DATA SCIENCE ACADEMY

1. Initialization

…in `lenet_in_tensorflow.ipynb`:

Set neural network hyperparameters

```
epochs = 20
batch_size = 128
display_progress = 40 # after this many batches, output progress to screen
wt_init = tf.contrib.layers.xavier_initializer() # weight initializer
```

```
weight_dict = {
    'W_c1': tf.get_variable('W_c1',
                            [k_conv_1, k_conv_1, 1, n_conv_1], initializer=wt_init),
    'W_c2': tf.get_variable('W_c2',
                            [k_conv_2, k_conv_2, n_conv_1, n_conv_2], initializer=wt_init),
    'W_d1': tf.get_variable('W_d1',
                            [dense_inputs, n_dense], initializer=wt_init),
    'W_out': tf.get_variable('W_out',
                             [n_dense, n_classes], initializer=wt_init)
}
```

2. Get Above Chance

If your accuracy is below chance, try:

- simplifying the problem
- simplifying the network architecture
- reducing your training set size (to iterate more quickly)

2. Get Above Chance

If your accuracy is below chance, try:

- simplifying the problem
- simplifying the network architecture
- reducing your training set size (to iterate more quickly)

2. Get Above Chance

If your accuracy is below chance, try:

- simplifying the problem
- simplifying the network architecture
- reducing your training set size (to iterate more quickly)

3. Layers

Experiment with varying:

- number of layers
- type of layers
- layer width (by powers of two)

3. Layers

Experiment with varying:

- number of layers
- type of layers
- layer width (by powers of two)

3. Layers

Experiment with varying:

- number of layers
- type of layers
- layer width (by powers of two)

4. Cost

…in `lenet_in_keras.ipynb`:

Configure model

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

…in `lenet_in_tensorflow.ipynb`:

Define model's loss and its optimizer

```
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=predictions, labels=y))
optimizer = tf.train.AdamOptimizer().minimize(cost)
```

5. Avoid Overfitting

If validation cost begins to increase or validation accuracy begins to decrease, consider:

- stopping training earlier
- dropout

5. Avoid Overfitting

If validation cost begins to increase or validation accuracy begins to decrease, consider:

- stopping training earlier
- dropout

5. Avoid Overfitting

...in `lenet_in_keras.ipynb`:

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(n_classes, activation='softmax'))
```

...in `lenet_in_tensorflow.ipynb`:

```
# max pooling layer:
pool_size = 2
mp_layer_dropout = 0.25

# dense layer:
n_dense = 128
dense_layer_dropout = 0.5
```

```
# convolutional and max-pooling layers:
conv_1 = conv2d(square_x, weights['W_c1'], biases['b_c1'])
conv_2 = conv2d(conv_1, weights['W_c2'], biases['b_c2'])
pool_1 = maxpooling2d(conv_2, mp_psize)
pool_1 = tf.nn.dropout(pool_1, 1-mp_dropout)

# dense layer:
flat = tf.reshape(pool_1, [-1, weights['W_d1'].get_shape().as_list()[0]])
dense_1 = dense(flat, weights['W_d1'], biases['b_d1'])
dense_1 = tf.nn.dropout(dense_1, 1-dense_dropout)
```

6. Learning Rate

...in `lenet_in_keras.ipynb`:

Configure model

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

...in `lenet_in_tensorflow.ipynb`:

Define model's loss and its optimizer

```
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=predictions, labels=y))
optimizer = tf.train.AdamOptimizer().minimize(cost)
```

7. Epochs

...in `lenet_in_keras.ipynb`:

Train!

```
model.fit(X_train, y_train, batch_size=128, epochs=20, verbose=1, validation_data=(X_test, y_test))
```

...in `lenet_in_tensorflow.ipynb`:

Set neural network hyperparameters

```
epochs = 20
batch_size = 128
display_progress = 40 # after this many batches, output progress to screen
wt_init = tf.contrib.layers.xavier_initializer() # weight initializer
```

```
with tf.Session() as session:
    session.run(initializer_op)

    print("Training for", epochs, "epochs.")

    # loop over epochs:
    for epoch in range(epochs):
```

8. Regularization λ

If using L1 or L2 regularization, consider:

- adjusting λ by orders of magnitude

9. Batch Size

...in `lenet_in_keras.ipynb`:

Train!

```
model.fit(X_train, y_train, batch_size=128, epochs=20, verbose=1, validation_data=(X_test,
y_test))
```

...in `lenet_in_tensorflow.ipynb`:

Set neural network hyperparameters

```
epochs = 20
batch_size = 128
display_progress = 40 # after this i
wt_init = tf.contrib.layers.xavier_i
```

```
# loop over all batches of the epoch:
n_batches = int(mnist.train.num_examples / batch_size
for i in range(n_batches):

    # to reassure you something's happening!
    if i % display_progress == 0:
        print("Step ", i+1, " of ", n_batches, " in epoch ", epoch+1, ".", sep='')

    batch_x, batch_y = mnist.train.next_batch(batch_size
```

10. Automation

For grid search of hyperparameters, consider:

- sampling values instead of looping over fixed values
- using [Spearmint]

10. Automation

For grid search of hyperparameters, consider:

- sampling values instead of looping over fixed values
- using [Spearmint]

All together now...

1. Initialization
2. Get Above Chance
3. Layers
4. Cost
5. Avoid Overfitting
6. Learning Rate
7. Epochs
8. Regularization λ
9. Batch Size
10. Automation

All together now...

1. Initialization
2. Get Above Chance
3. Layers
4. Cost
5. Avoid Overfitting
6. Learning Rate
7. Epochs
8. Regularization λ
9. Batch Size
10. Automation

All together now...

1. Initialization
2. Get Above Chance
3. Layers
4. Cost
5. Avoid Overfitting
6. Learning Rate
7. Epochs
8. Regularization λ
9. Batch Size
10. Automation

All together now...

1. Initialization
2. Get Above Chance
3. Layers
4. Cost
5. Avoid Overfitting
6. Learning Rate
7. Epochs
8. Regularization λ
9. Batch Size
10. Automation

All together now...

1. Initialization
2. Get Above Chance
3. Layers
4. Cost
5. Avoid Overfitting
6. Learning Rate
7. Epochs
8. Regularization λ
9. Batch Size
10. Automation

All together now...

1. Initialization
2. Get Above Chance
3. Layers
4. Cost
5. Avoid Overfitting
6. Learning Rate
7. Epochs
8. Regularization λ
9. Batch Size
10. Automation

All together now...

1. Initialization
2. Get Above Chance
3. Layers
4. Cost
5. Avoid Overfitting
6. Learning Rate
7. Epochs
8. Regularization λ
9. Batch Size
10. Automation

All together now...

1. Initialization
2. Get Above Chance
3. Layers
4. Cost
5. Avoid Overfitting
6. Learning Rate
7. Epochs
8. Regularization λ
9. Batch Size
10. Automation

All together now...

1. Initialization
2. Get Above Chance
3. Layers
4. Cost
5. Avoid Overfitting
6. Learning Rate
7. Epochs
8. Regularization λ
9. Batch Size
10. Automation

All together now...

1. Initialization
2. Get Above Chance
3. Layers
4. Cost
5. Avoid Overfitting
6. Learning Rate
7. Epochs
8. Regularization λ
9. Batch Size
10. Automation

Outline

NYC DATA SCIENCE ACADEMY

Generative Adversarial Networks

April 7th —*In a Fortnight!*

Reinforcement Learning

April 7th —*In a Fortnight!*