



# Convnets in TensorFlow

CS 20: TensorFlow for Deep Learning Research

Lecture 7

2/7/2017



# Agenda

Convolutions without training

Convnet with MNIST!!!

`tf.layers`



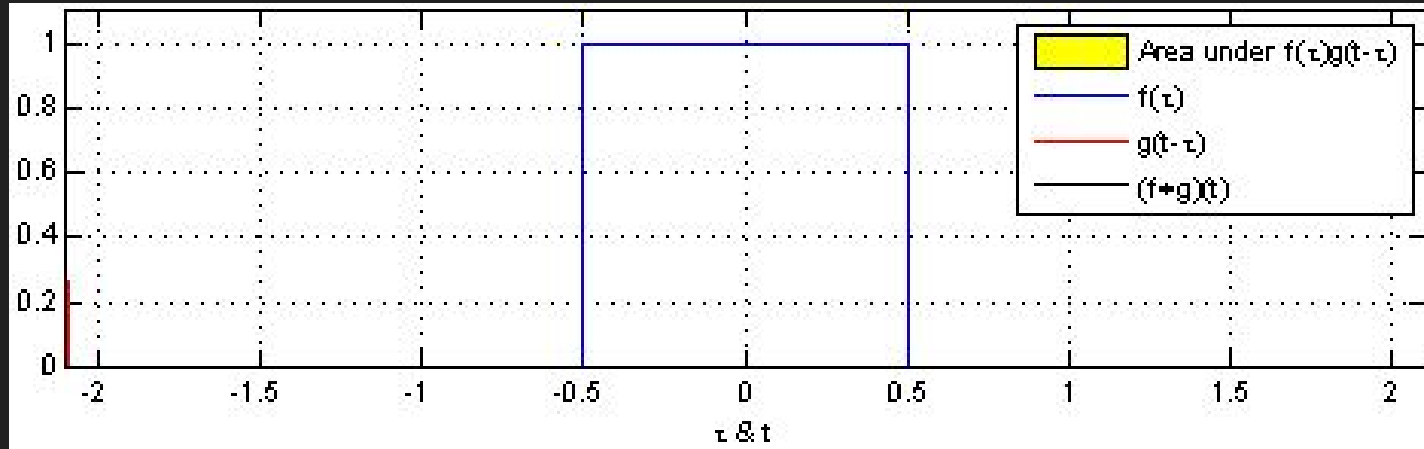


# Understanding convolutions

# Convolutions in math and physics

a function derived from two given functions by integration that expresses how the shape of one is modified by the other

# Convolutions in math and physics



# Convolutions in math and physics

How an input is transformed by a kernel\*

\*also called filter/feature map

# Convolutions in machine learning

We can use one single convolutional layer to modify a certain image



# Convolutions in machine learning

1 <sub>x1</sub>	1 <sub>x0</sub>	1 <sub>x1</sub>	0	0
0 <sub>x0</sub>	1 <sub>x1</sub>	1 <sub>x0</sub>	1	0
0 <sub>x1</sub>	0 <sub>x0</sub>	1 <sub>x1</sub>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved  
Feature

# Kernel for blurring

0.0625	0.125	0.0625
0.125	0.25	0.125
0.0625	0.125	0.0625

Matrix multiplication of this kernel with  
a 3 x 3 patch of an image is a weighted sum  
of neighboring pixels  
=> blurring effect

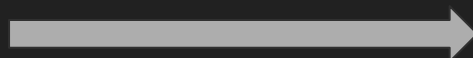
# Convolution without training



input

Kernel for blurring

0.0625	0.125	0.0625
0.125	0.25	0.125
0.0625	0.125	0.0625



`tf.nn.conv2d`



output

# Convolutions in TensorFlow

We can use one single convolutional layer to modify a certain image

```
tf.nn.conv2d(  
    input,  
    filter,  
    strides,  
    padding,  
    use_cudnn_on_gpu=True,  
    data_format='NHWC',  
    dilations=[1, 1, 1, 1],  
    name=None  
)
```

# Convolutions in TensorFlow

We can use one single convolutional layer to modify a certain image

```
tf.nn.conv2d(  
    input,           Batch size (N) x Height (H) x Width (W) x Channels (C)  
    filter,          Height x Width x Input Channels x Output Channels  
    strides,         4 element 1-D tensor, strides in each direction  
    padding,         'SAME' or 'VALID'  
    use_cudnn_on_gpu=True,  
    data_format='NHWC',  
    dilations=[1, 1, 1, 1],  
    name=None  
)
```

# Convolutions in TensorFlow

We can use one single convolutional layer to modify a certain image

```
tf.nn.conv2d(  
    image,  
    kernel,  
    strides=[1, 3, 3, 1],  
    padding='SAME',  
)
```



# Some basic kernels



input



sharpen



edge



top sobel



emboss

See `kernels.py` and `07_run_kernels.py`

# Some basic kernels



input



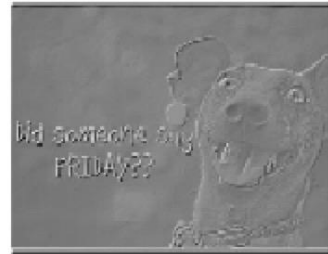
sharpen



edge



top sobel



emboss





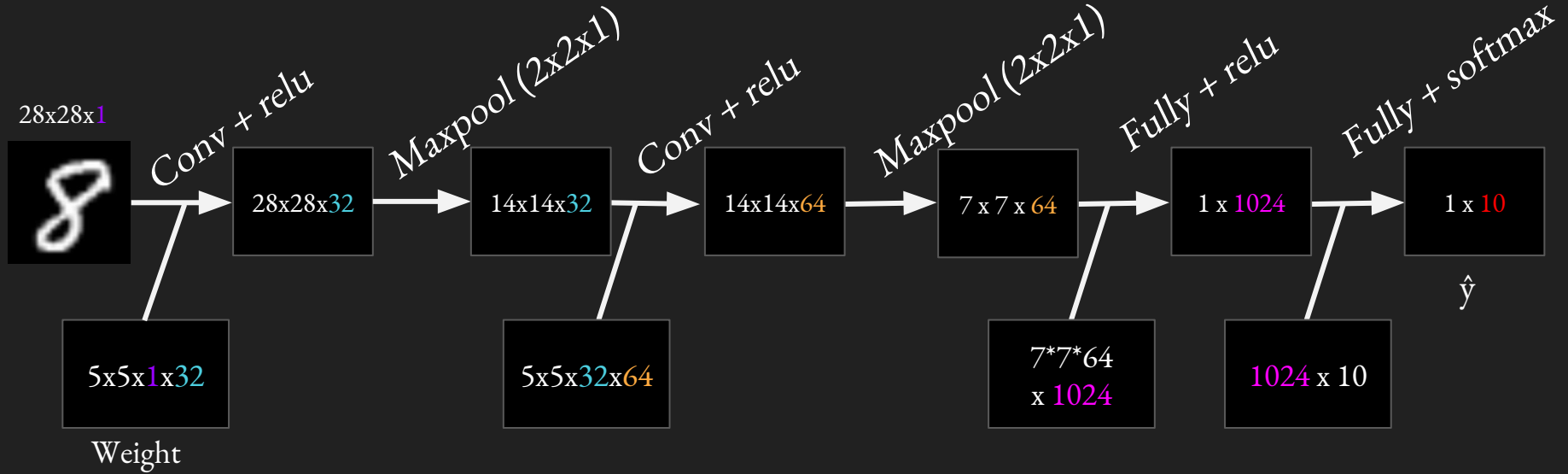
# Convolutions in machine learning

Don't hard-code the values of your kernels.  
Learn the optimal kernels through training!



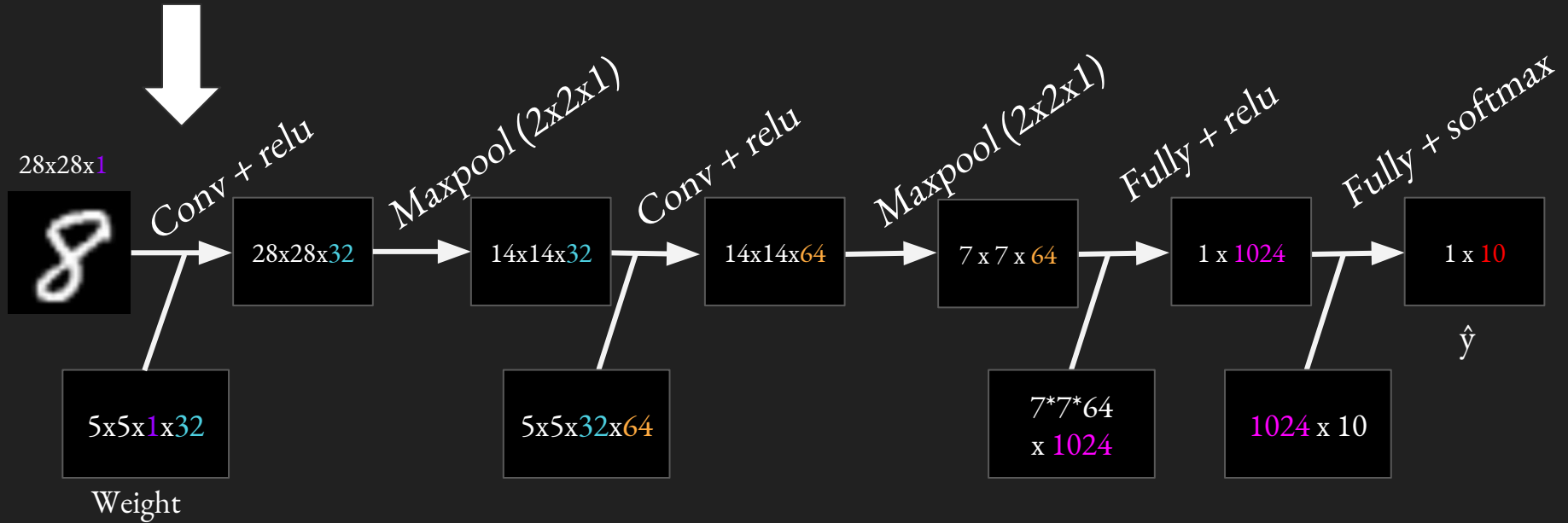
# ConvNet with MNIST

# Model



Strides for all convolutional layers: [1, 1, 1, 1]

# Convolutional layer



```
conv = tf.nn.conv2d(images,  
                    kernel,  
                    strides=[1, 1, 1, 1],  
                    padding='SAME')
```

# Convolutional layer: padding

"VALID" = without padding:

inputs:

	1	2	3	4	5	6	7	8	9	10	11	(12 13)
		_____					_____					dropped

"SAME" = with zero padding:

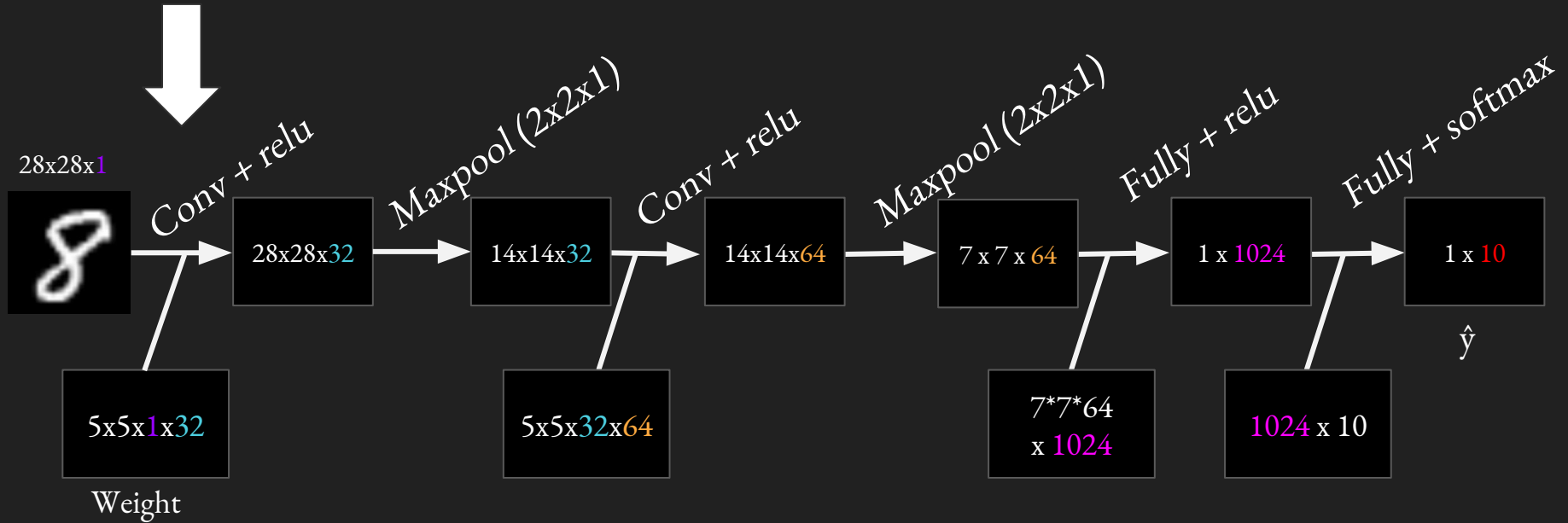
[illegible]

Input width = 13

Filter width = 6

Stride = 5

# Convolutional layer: Dimension

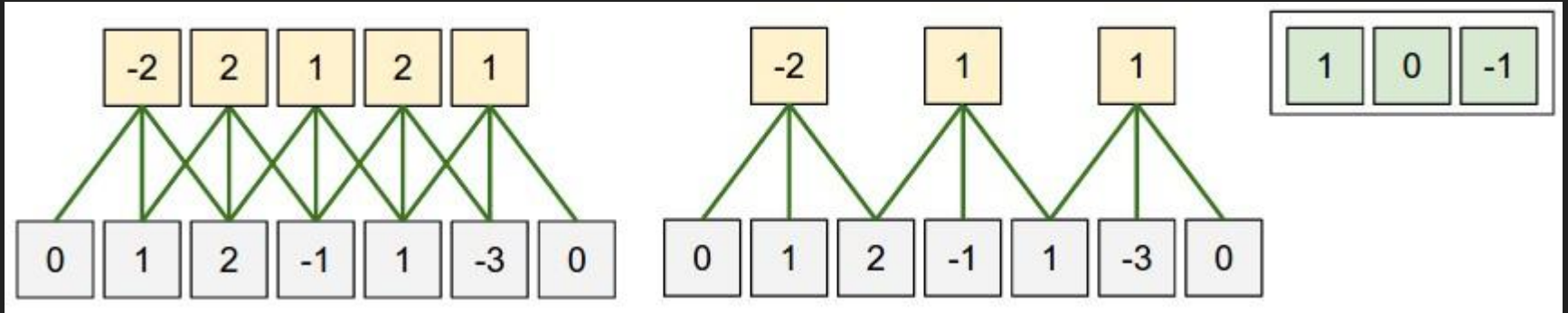


$$(W - F + 2P) / S + 1$$

W: input width/depth  
P: padding

F: filter width/depth  
S: stride

# Convolutional layer: Dimension

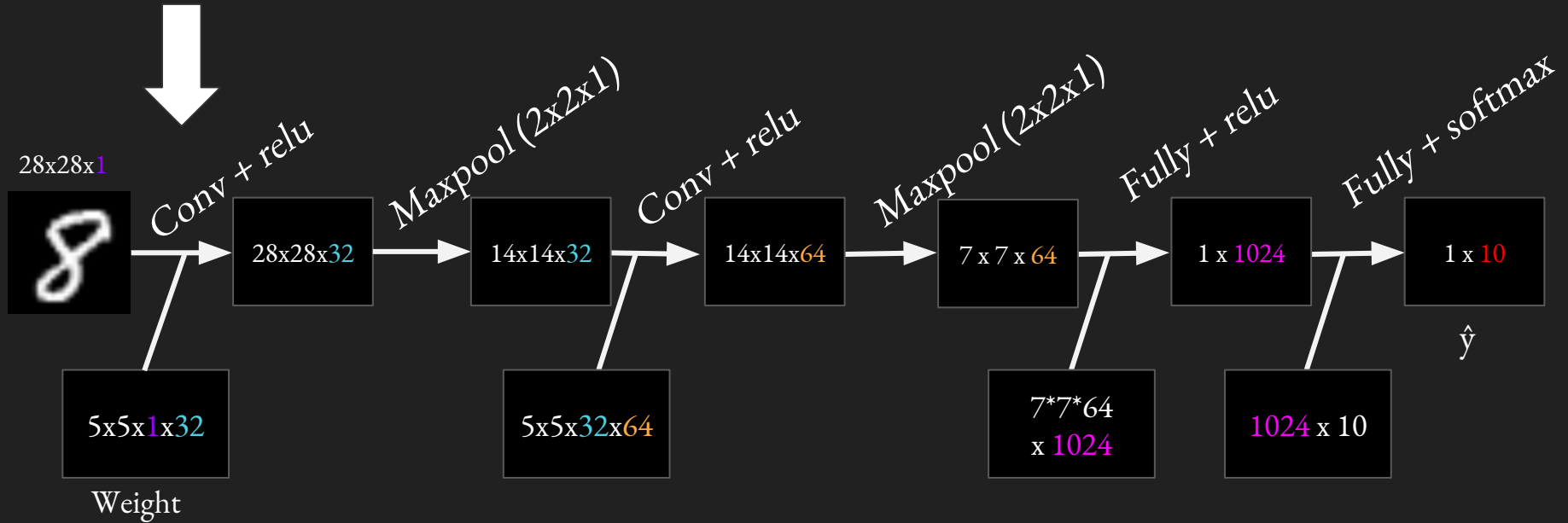


$$(W - F + 2P) / S + 1$$

W: input width/depth  
P: padding

F: filter width/depth  
S: stride

# Convolutional layer: Dimension



$$(\mathbf{W}-\mathbf{F}+2\mathbf{P})/\mathbf{S}+1$$

$$(28-5+2*2)/1+1=28$$

W: input width/depth

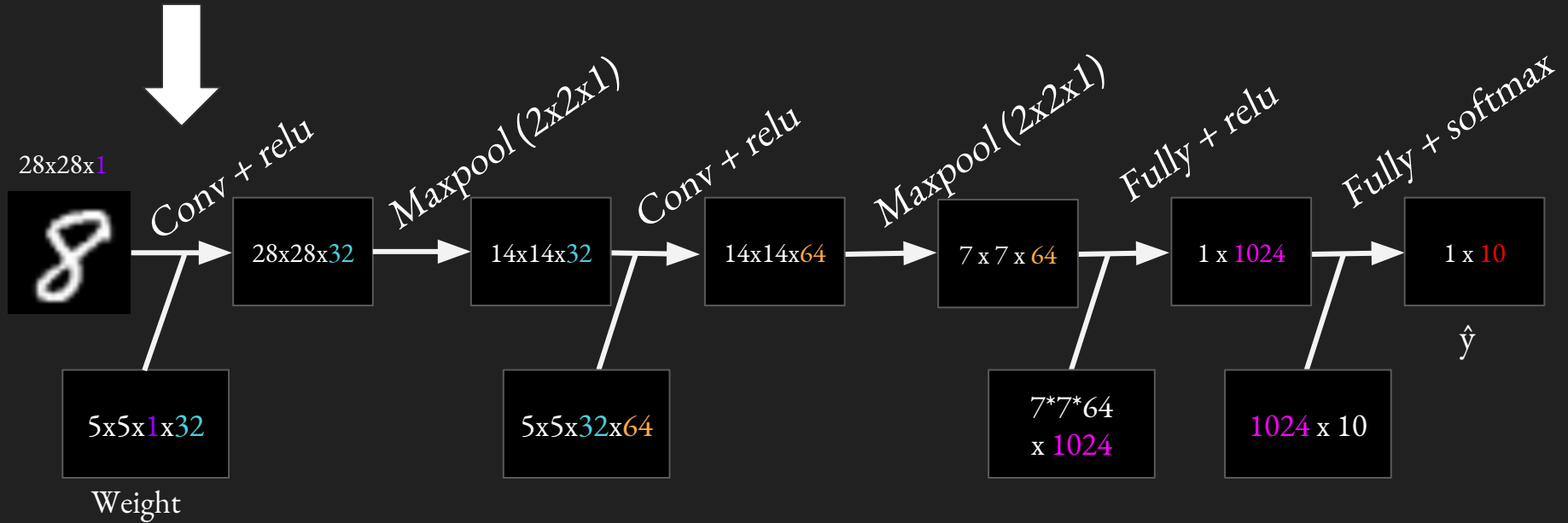
P: padding

F: filter width/depth

S: stride



# Convolutional layer: Dimension



$$(\mathbf{W} - \mathbf{F} + 2\mathbf{P}) / \mathbf{S} + 1$$

$$(28 - 5 + 2 * 2) / 1 + 1 = 28$$

W: input width/depth

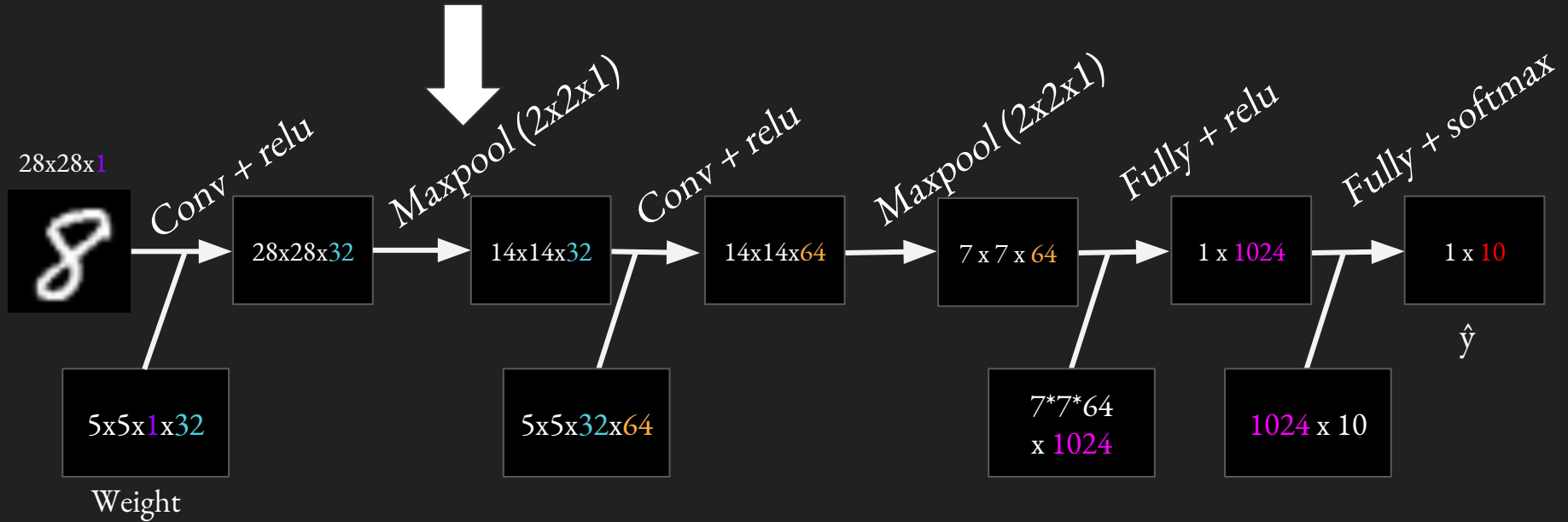
P: padding

F: filter width/depth

S: stride

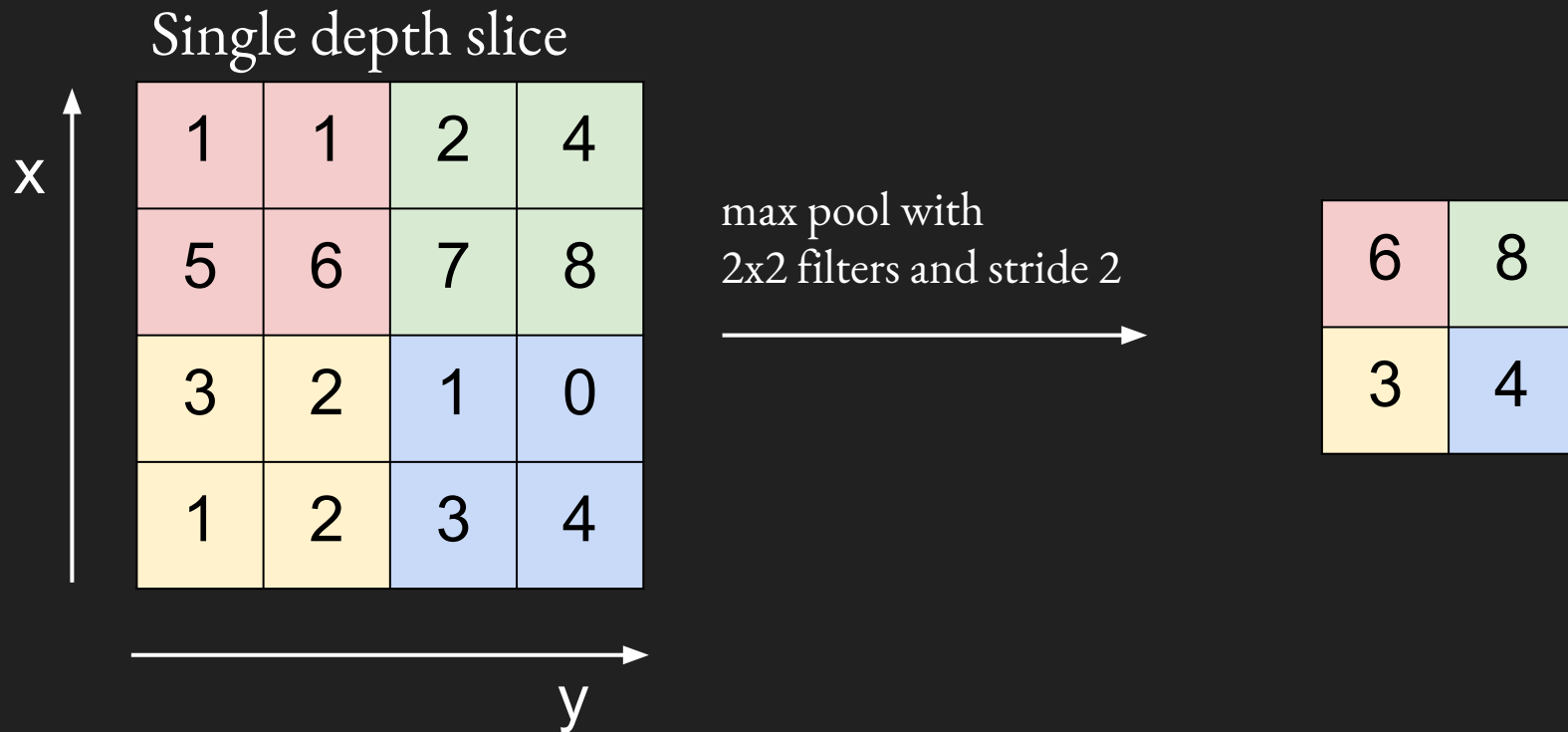
TF computes padding for us!

# Maxpooling

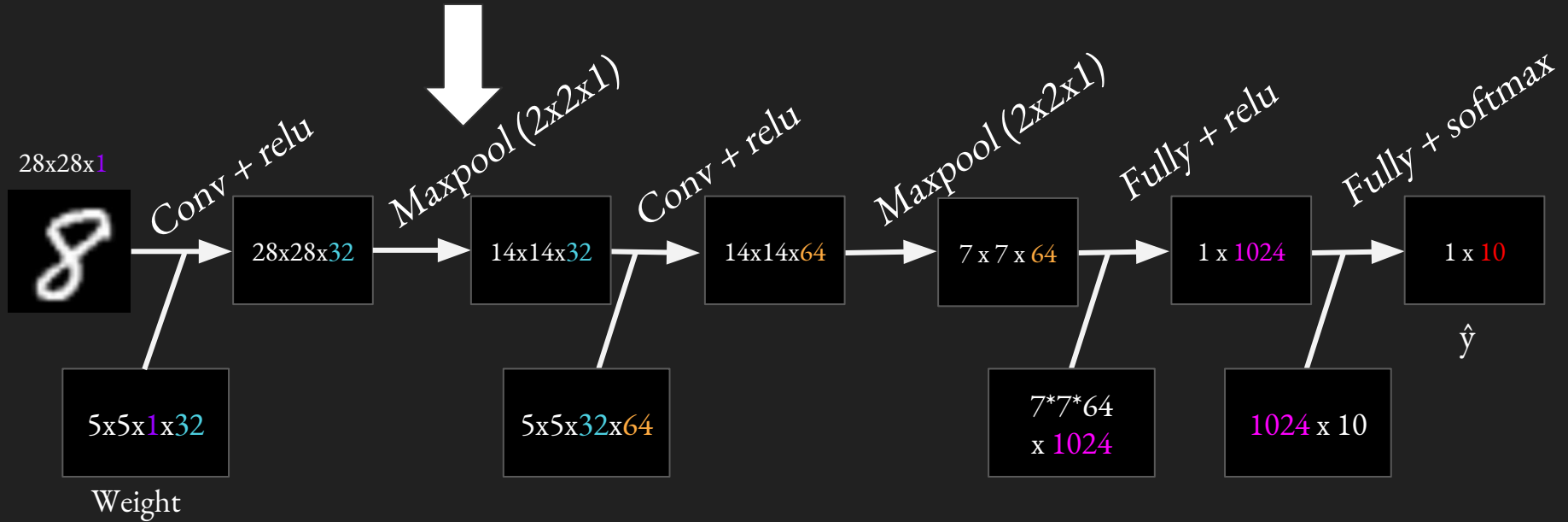


```
pool1 = tf.nn.max_pool(conv1,  
                        ksize=[1, 2, 2, 1],  
                        strides=[1, 2, 2, 1],  
                        padding='SAME')
```

# Maxpooling



# Maxpooling: Dimension

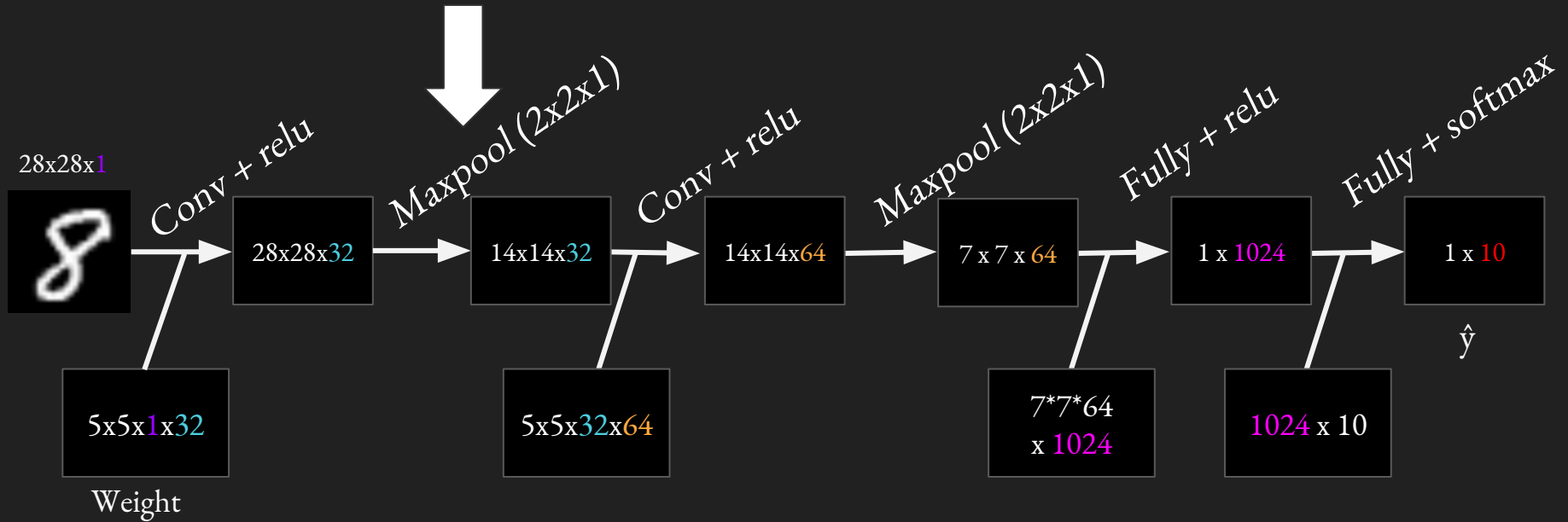


$$(W-K+2P)/S+1$$

W: input width/depth  
P: padding

K: window width/depth  
S: stride

# Maxpooling: Dimension



$$(\mathbf{W} - \mathbf{K} + 2\mathbf{P}) / \mathbf{S} + 1$$

$$(28 - 2 + 2 \times 0) / 2 + 1 = 14$$

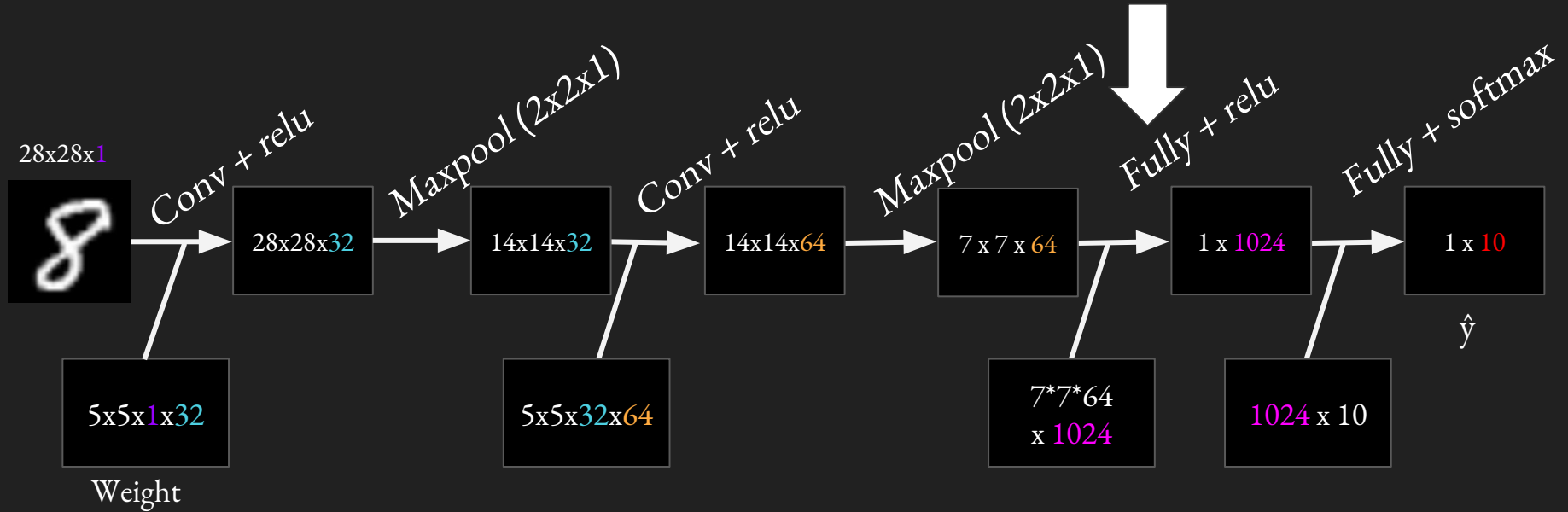
W: input width/depth

P: padding

K: window width/depth

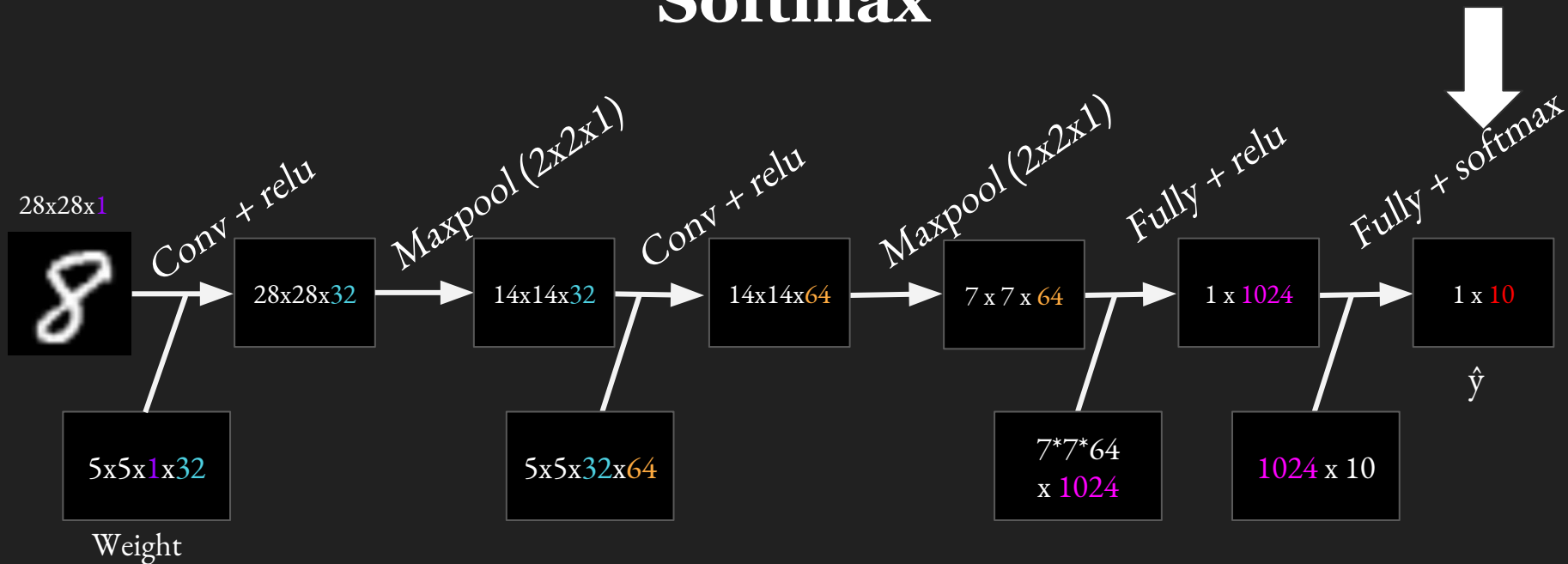
S: stride

# Fully connected



`fc = tf.matmul(pool2, w) + b`

# Softmax



Loss function

`tf.nn.softmax_cross_entropy_with_logits(labels=Y, logits=logits)`

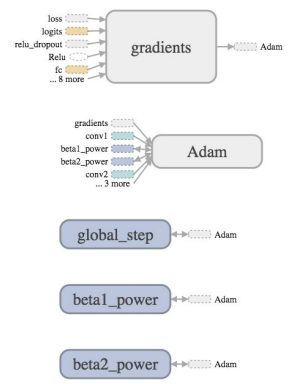
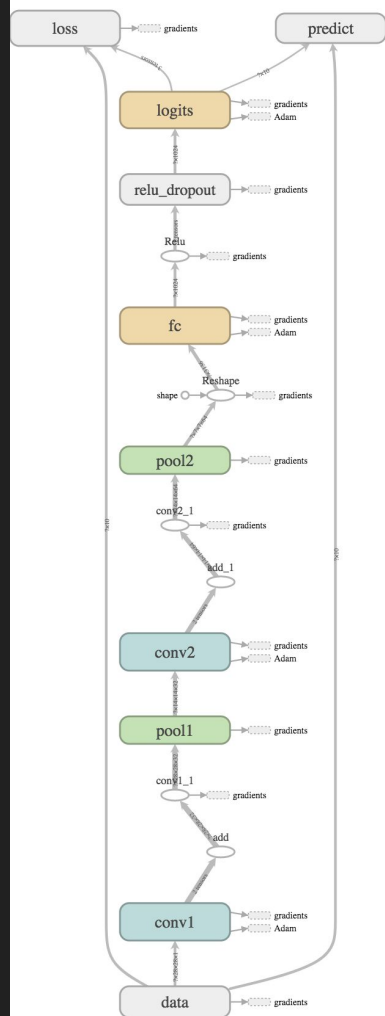
Predict

`tf.nn.softmax(logits_batch)`

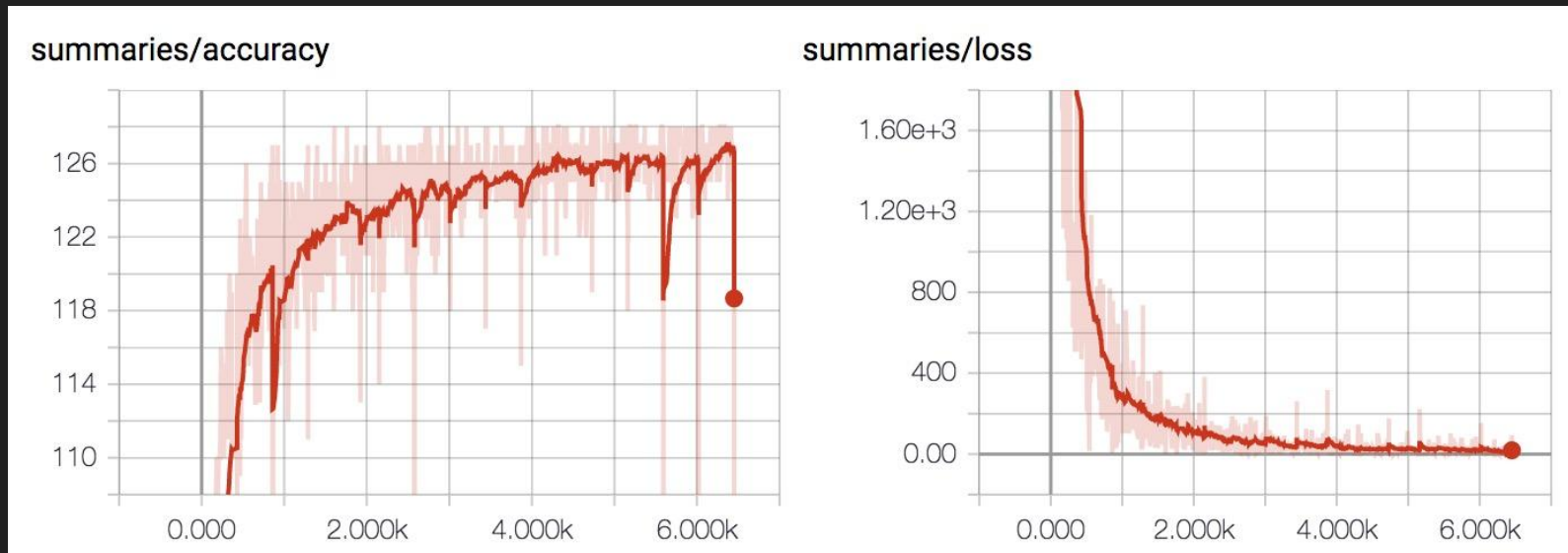
# Interactive coding

07\_convnet\_mnist\_starter.py from GitHub!  
Update utils.py





# Training progress



Test accuracy increases while training loss decreases!

# Accuracy

Epochs	Accuracy
1	0.9131
2	0.9363
3	0.9478
5	0.9573
10	0.971
25	0.9818



tf.layers

# tf.layers

We've been learning it the hard way

# tf.layers.conv2d

```
conv1 = tf.layers.conv2d(inputs=self.img,  
                          filters=32,  
                          kernel_size=[5, 5],  
                          padding='SAME',  
                          activation=tf.nn.relu,  
                          name='conv1')
```

# tf.layers.conv2d

```
conv1 = tf.layers.conv2d(inputs=self.img,  
                          filters=32,  
                          kernel_size=[5, 5],  
                          padding='SAME',  
                          activation=tf.nn.relu,  
                          name='conv1')
```

can choose  
non-linearity to use

# **tf.layers.max\_pooling2d**

```
pool1 = tf.layers.max_pooling2d(inputs=conv1,  
                                pool_size=[2, 2],  
                                strides=2,  
                                name='pool1')
```



# **tf.layers.dense**

```
fc = tf.layers.dense(pool2, 1024, activation=tf.nn.relu, name='fc')
```

# tf.layers.dense

```
dropout = tf.layers.dropout(fc,  
                             self.keep_prob,  
                             training=self.training,  
                             name='dropout')
```

Drop neurals during training  
Want to use all of them during testing

# Next class

TFRecord

CIFAR

Style Transfer

Feedback: [chipuyen@cs.stanford.edu](mailto:chipuyen@cs.stanford.edu)

Thanks!