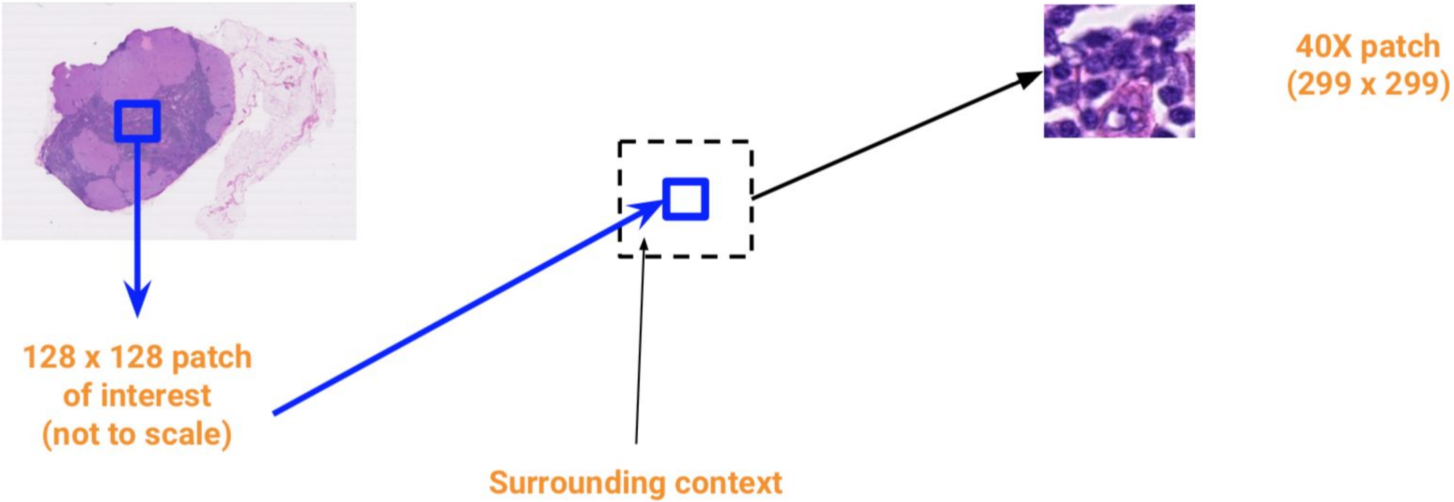
Applied Deep Learning

Midterm Review · Mar 7th, 2019

Question: In “Detecting Cancer Metastases on Gigapixel Pathology Images”, the authors used a CNN-based model to detect and localize tumors in large images. Instead of training a single CNN-based model, the authors chose to train four, and used their output to arrive at a final prediction. Why?

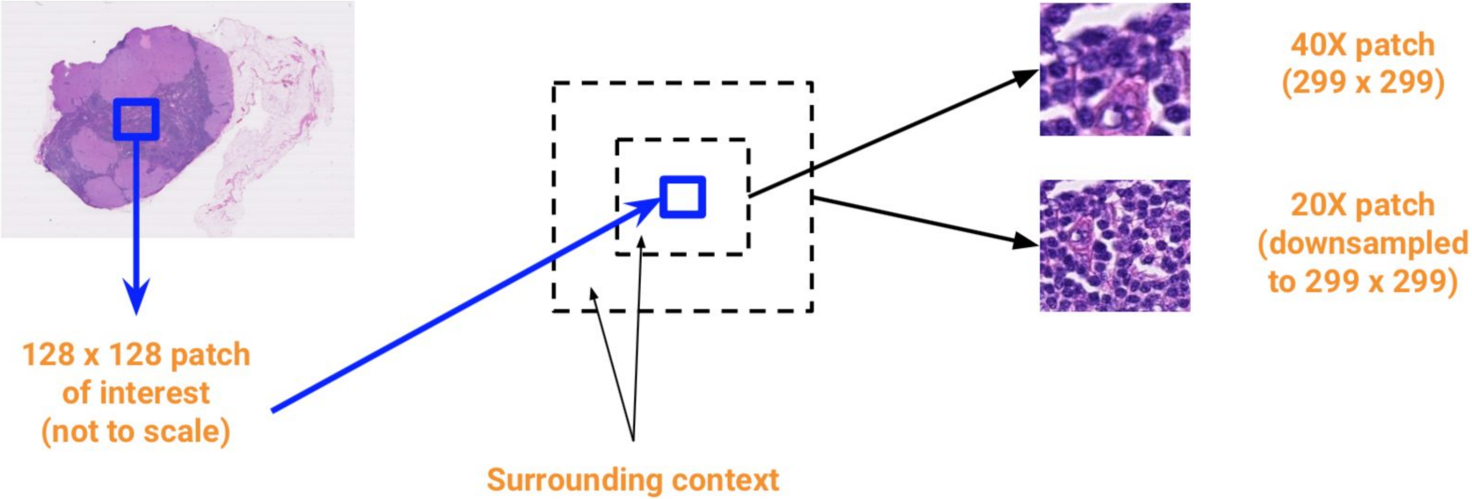


40X patch
(299 x 299)

128 x 128 patch
of interest
(not to scale)

Surrounding context

Detecting Cancer Metastases on Gigapixel Pathology Images, 2017



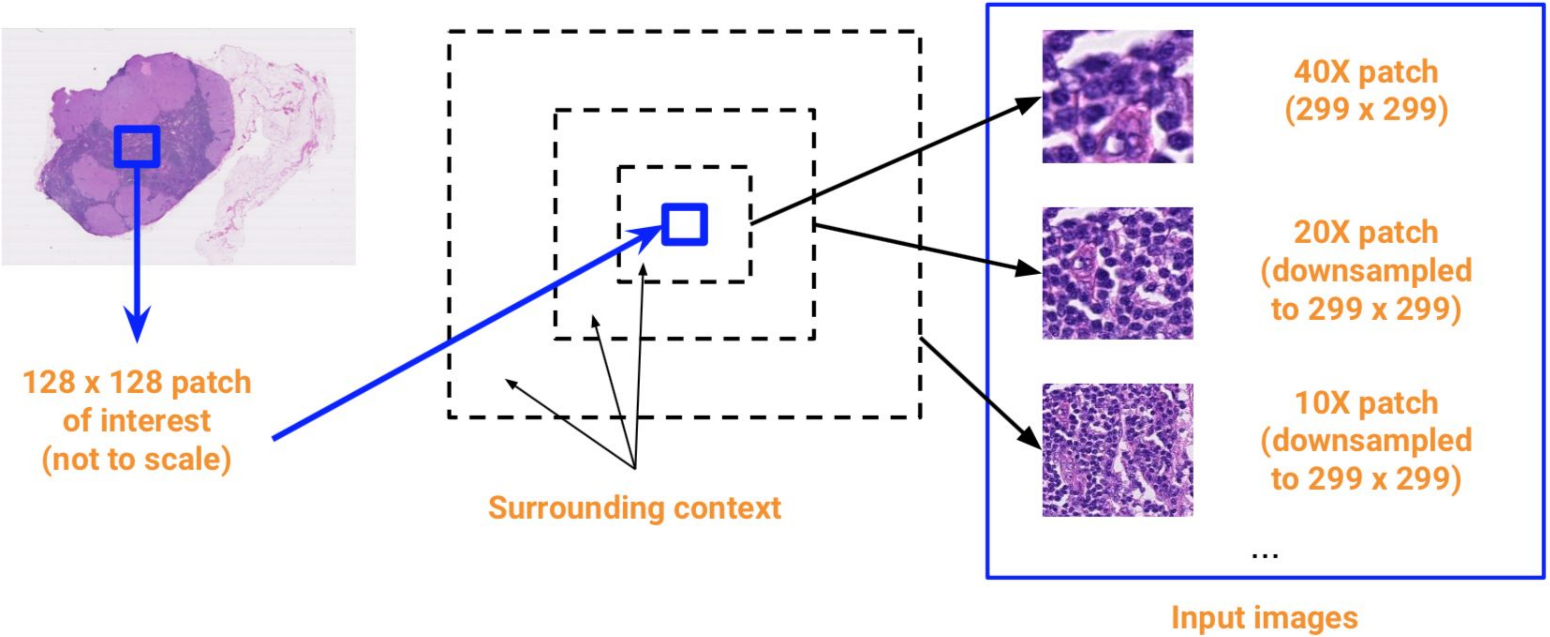
128 x 128 patch of interest (not to scale)

Surrounding context

40X patch (299 x 299)

20X patch (downsampled to 299 x 299)

Detecting Cancer Metastases on Gigapixel Pathology Images, 2017



128 x 128 patch of interest (not to scale)

Surrounding context

40X patch (299 x 299)

20X patch (downsampled to 299 x 299)

10X patch (downsampled to 299 x 299)

...

Input images

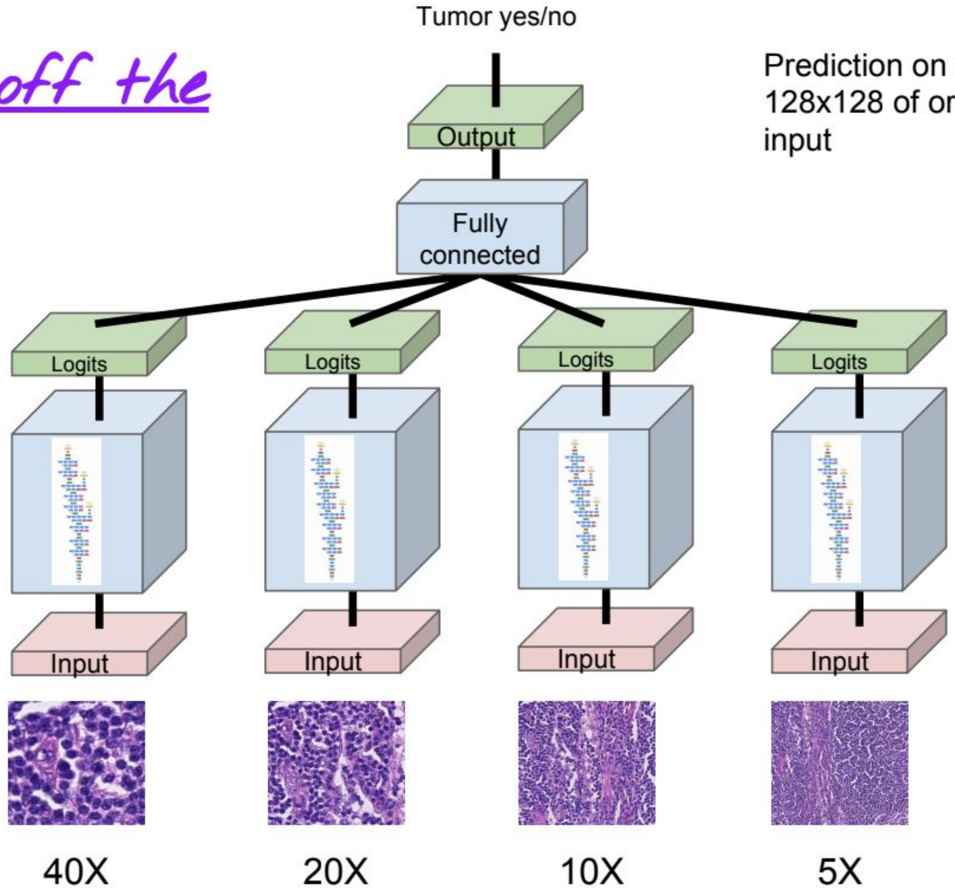
Detecting Cancer Metastases on Gigapixel Pathology Images, 2017

Model is mostly off the shelf!

Multi scale model

detail ←→ context

resembles microscope magnifications



Prediction on center 128x128 of original 40X input

Independent Inception v3 towers

299x299 input patches at different scales

Detecting Cancer Metastases on Gigapixel Pathology Images, 2017

Answer: to capture detail as well as context (the authors needed a way to include multiple zoom levels in the training data). Solution was to train four models (at different zoom levels), and use a dense layer on top to weight the results.

Question: What is the purpose of data augmentation?

Question: Should you use data augmentation for the training set, the validation set and/or the test set?

Less obvious transforms may be relevant for your domain

Data augmentation may be applied to the **training set only**. Never to validation or test.

Computationally expensive (increasing the size of your training set by a factor of n_augmentations).

Most libraries provide data augmentation tools. I'll try to cover tf.data in lecture 6 to show how to do this at scale, though 99% of the time simpler options are more than adequate and your best bet.

https://keras.io/preprocessing/image/

Random jpeg quality

random_jpeg_quality

Answer: just use augmentation for the training set

Test - nothing to do with the training process

Answer: just use augmentation for the training set

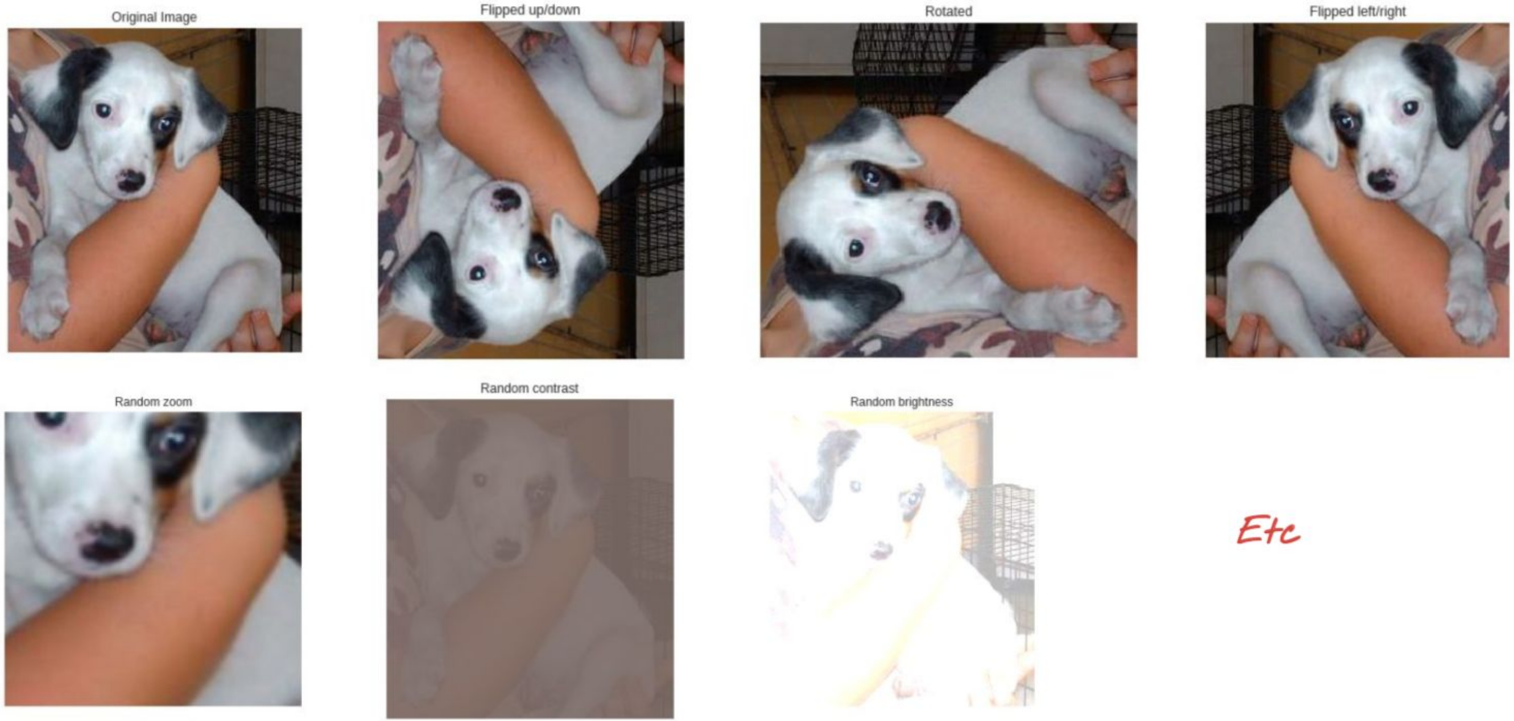
Test - nothing to do with the training process

Validation - should look like real-world data

data augmentation introduces distortions

Question: What kinds of augmentations might you apply?

Data augmentation



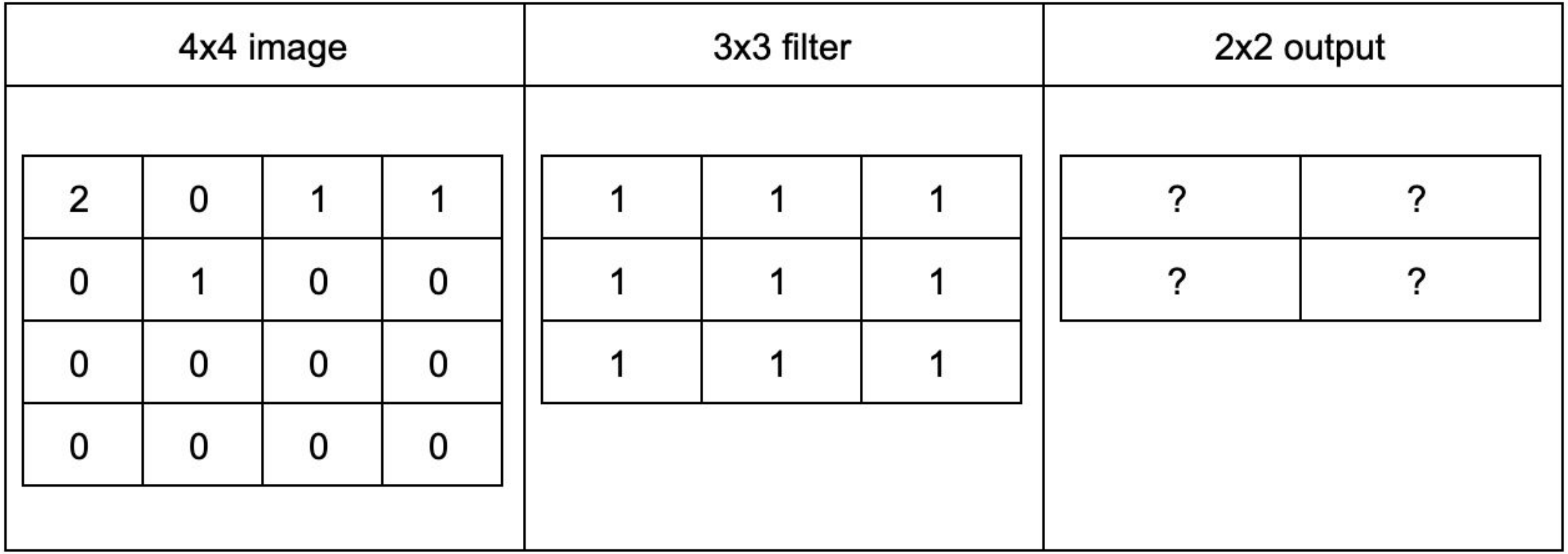
Etc

Question: What is the approximate increase in training time as a function of the number of augmentations you choose?

Question: What is the approximate increase in training time as a function of the number of augmentations you choose?

Answer: linear

Question: What is the result after convolving this 3x3 filter across this 4x4 image? Assume stride 1 and no padding.



<table>
<tr><th>4x4 image</th><th>3x3 filter</th><th>2x2 output</th></tr>
<tr>
<td><table>
<tr><td>2</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table></td>
<td><table>
<tr><td>1</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table></td>
<td><table>
<tr><td>?</td><td>?</td></tr>
<tr><td>?</td><td>?</td></tr>
</table></td>
</tr>
</table>

Example

Of course, in a CNN these filters are learned

2	0	1	1
0	1	0	0
0	0	1	0
0	3	0	0

An input image
(no padding)

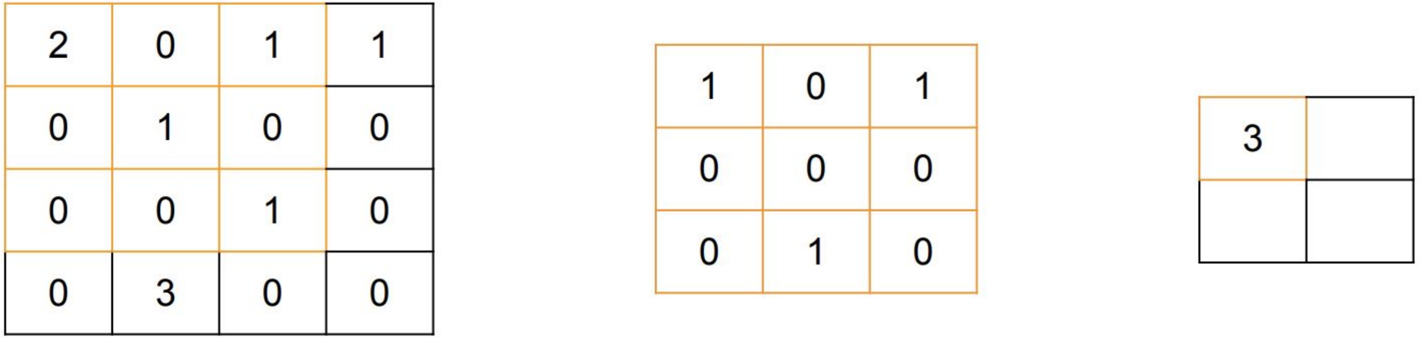
1	0	1
0	0	0
0	1	0

A filter
(3x3)

Output image
(after convolving with stride 1)

Example

Of course, in a CNN these filters are learned



2	0	1	1
0	1	0	0
0	0	1	0
0	3	0	0

An input image
(no padding)

1	0	1
0	0	0
0	1	0

A filter
(3x3)

3	

Output image
(after convolving with stride 1)

`np.tensordot(img[:3, :3], kernel)` or $2*1 + 0*0 + 1*1 + 0*0 + 1*0 + 0*0 + 0*0 + 0*1 + 1*0$

Example

Of course, in a CNN these filters are learned

2	0	1	1
0	1	0	0
0	0	1	0
0	3	0	0

An input image (no padding)

1	0	1
0	0	0
0	1	0

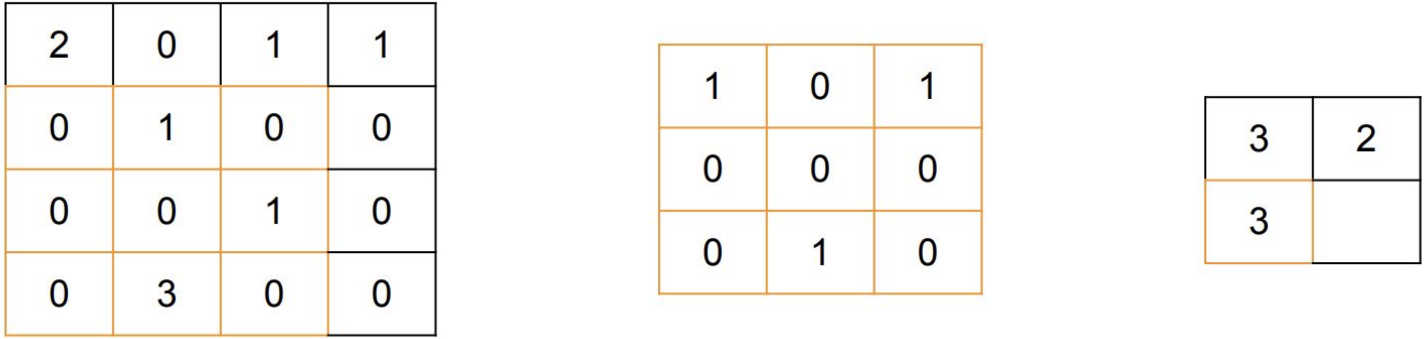
A filter (3x3)

3	2

Output image (after convolving with stride 1)

Example

Of course, in a CNN these filters are learned



2	0	1	1
0	1	0	0
0	0	1	0
0	3	0	0

An input image (no padding)

1	0	1
0	0	0
0	1	0

A filter (3x3)

3	2
3	

Output image (after convolving with stride 1)

Example

Of course, in a CNN these filters are learned

2	0	1	1
0	1	0	0
0	0	1	0
0	3	0	0

An input image
(no padding)

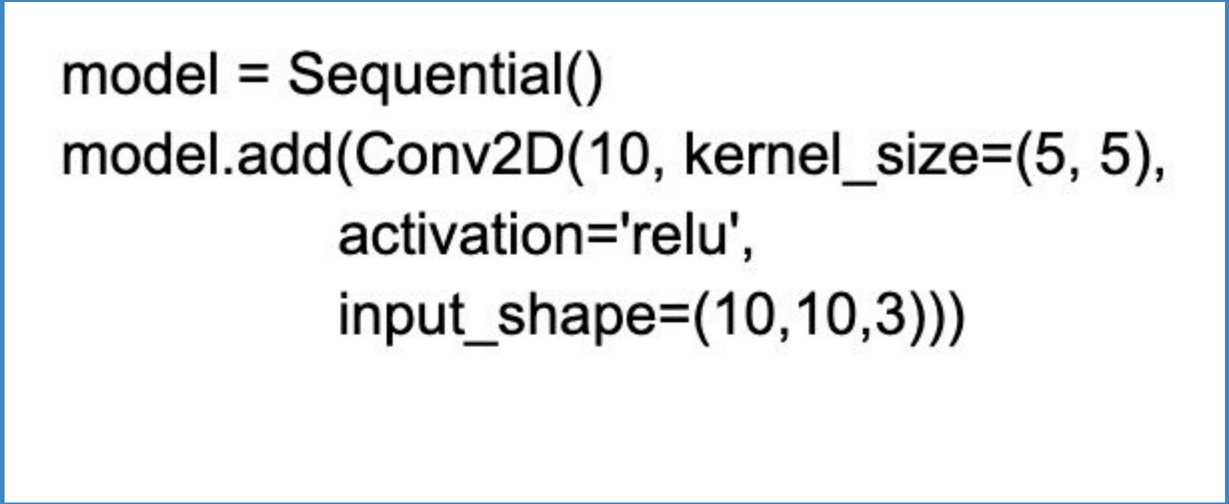
1	0	1
0	0	0
0	1	0

A filter
(3x3)

3	2
3	1

Output image
(after convolving with stride 1)

Question: How many parameters are in the convolutional layer below?



```
model = Sequential()
model.add(Conv2D(10, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=(10,10,3)))
```

Answer: 760

Parameters = (Filter Size) x (Input Depth) x (# of filters) + (bias, 1 per filter)

In our example,

Parameters = (5 x 5) x (3) x (10) + (10)

= 760

Question: What’s Dropout, and how does it work?

Answer: Dropout is used to drop neurons in the hidden layers of a model. It prevents overfitting.

The dropout rate is the fraction of the activations that are zeroed out. It is usually set between 0.2 and 0.5

It is important to remember that Dropout is only performed during training.

It reduces the learning capacity of the model and forces the model to learn redundant/important representations.

Question: Imagine a startup has contracted with the Department of Homeland Security to use facial recognition to identify potential criminals. What could go wrong? How would you approach this problem?

Answer: The data may be biased. So, it may wrongly classify normal people as criminals.

ACLU performed an experiment by training Amazon's 'Rekognition' model on publicly available mugshot photos and used the model to classify photos of members of the House and Senate. Nearly 40% of false matches in their test were of people of color.

Such kind of bias in the data needs to be taken care of while developing such an application.

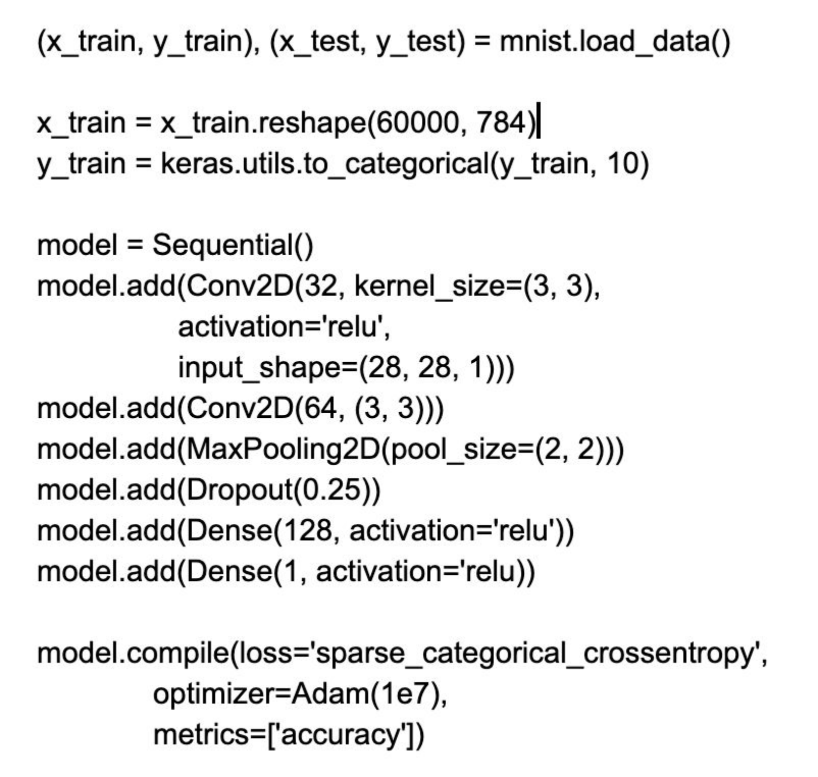
Make sure the data is diverse, by doing the following:

1. Data is evenly distributed across gender, age and ethnicity.
2. Include data which contains different poses, facial expressions, illuminations, occlusions and backgrounds.

Reference

Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification

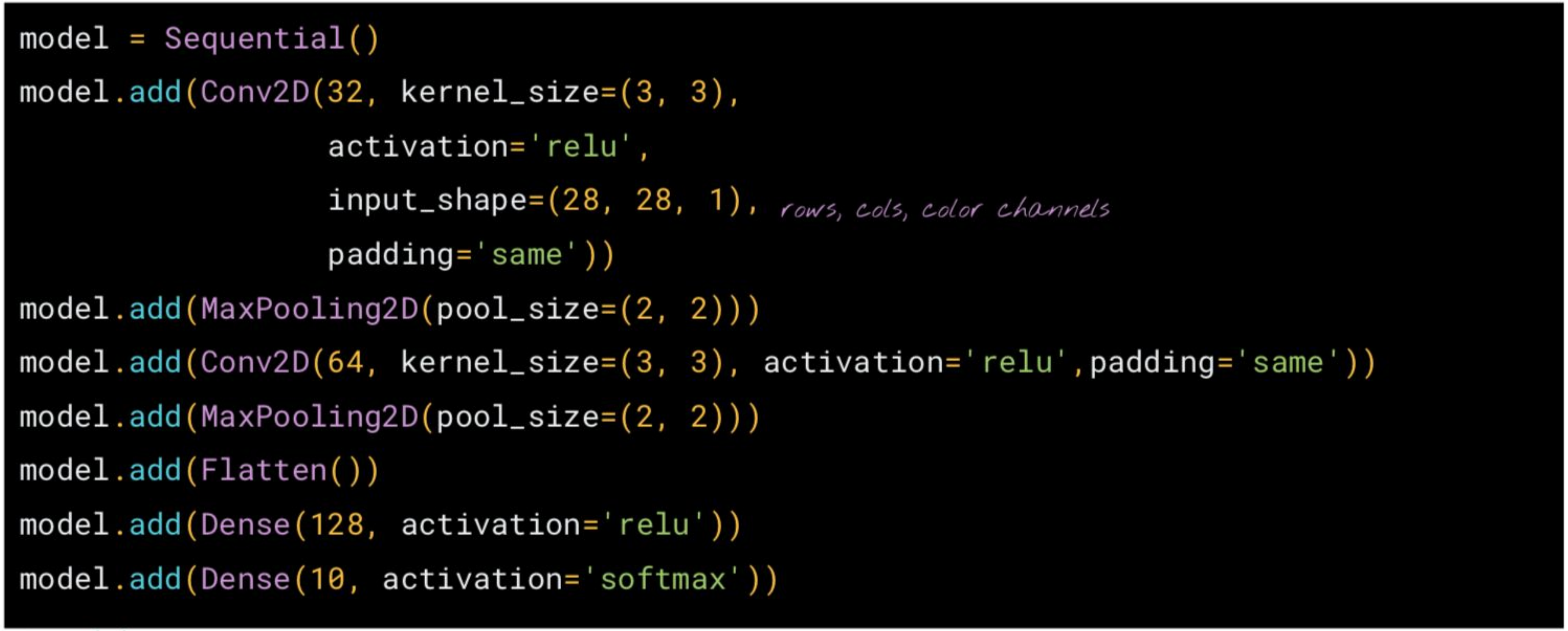
Question: A student has implemented their first CNN in Keras to classify MNIST digits. Please circle the bugs and write a one sentence description of each.



```
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(60000, 784)
y_train = keras.utils.to_categorical(y_train, 10)

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

model.compile(loss='sparse_categorical_crossentropy',
        optimizer=Adam(1e7),
        metrics=['accuracy'])
```

Example

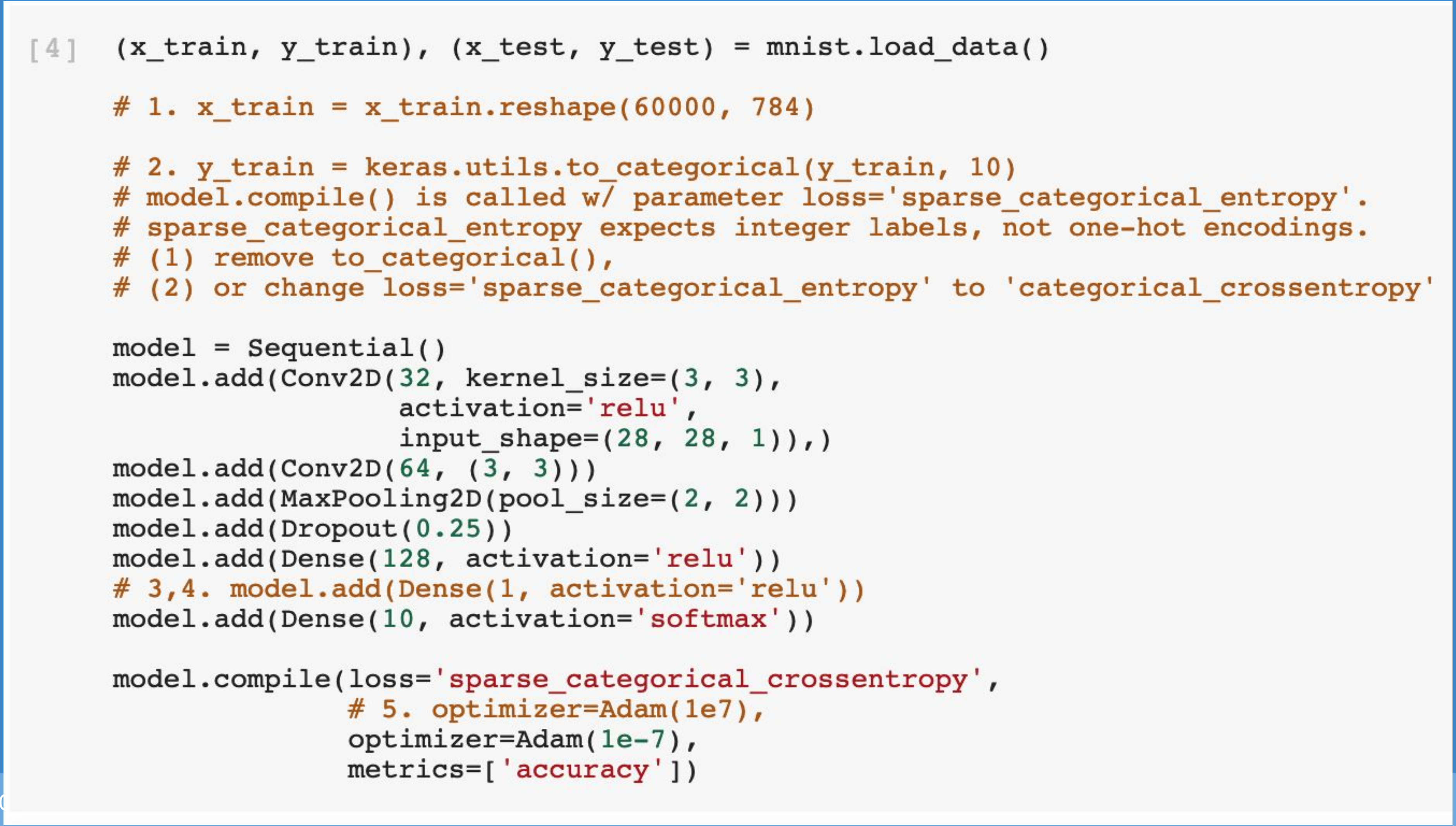
A typical CNN in Keras



```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=(28, 28, 1),
                 padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu',padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

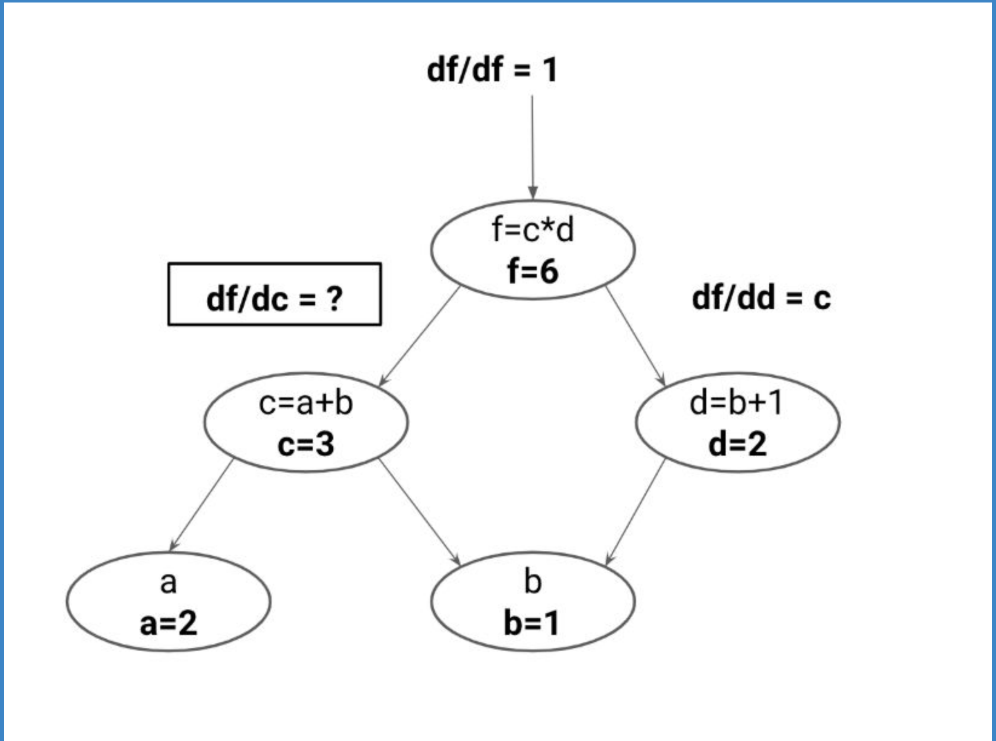
rows, cols, color channels

A good place to start

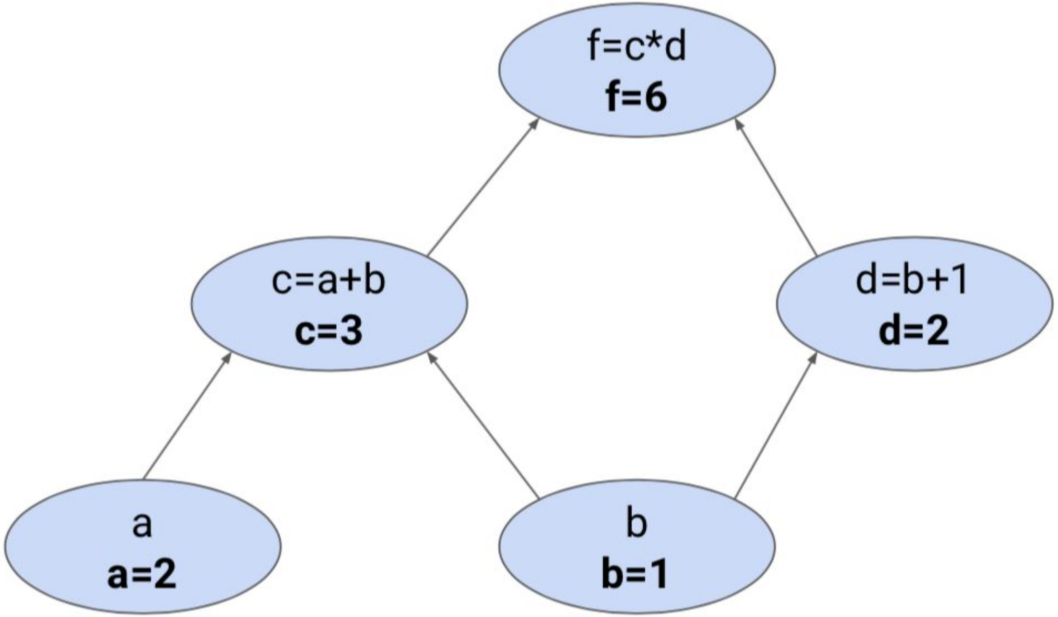


```
[4]  (x_train, y_train), (x_test, y_test) = mnist.load_data()

     # 1. x_train = x_train.reshape(60000, 784)

     # 2. y_train = keras.utils.to_categorical(y_train, 10)
     # model.compile() is called w/ parameter loss='sparse_categorical_entropy'.
     # sparse_categorical_entropy expects integer labels, not one-hot encodings.
     # (1) remove to_categorical(),
     # (2) or change loss='sparse_categorical_entropy' to 'categorical_crossentropy'

     model = Sequential()
     model.add(Conv2D(32, kernel_size=(3, 3),
                      activation='relu',
                      input_shape=(28, 28, 1)),)
     model.add(Conv2D(64, (3, 3)))
     model.add(MaxPooling2D(pool_size=(2, 2)))
     model.add(Dropout(0.25))
     model.add(Dense(128, activation='relu'))
     # 3,4. model.add(Dense(1, activation='relu'))
     model.add(Dense(10, activation='softmax'))

     model.compile(loss='sparse_categorical_crossentropy',
                   # 5. optimizer=Adam(1e7),
                   optimizer=Adam(1e-7),
                   metrics=['accuracy'])
```

Question: What does df/dc represent in the below diagram? Be specific if you can.



The gradient gives us the answer



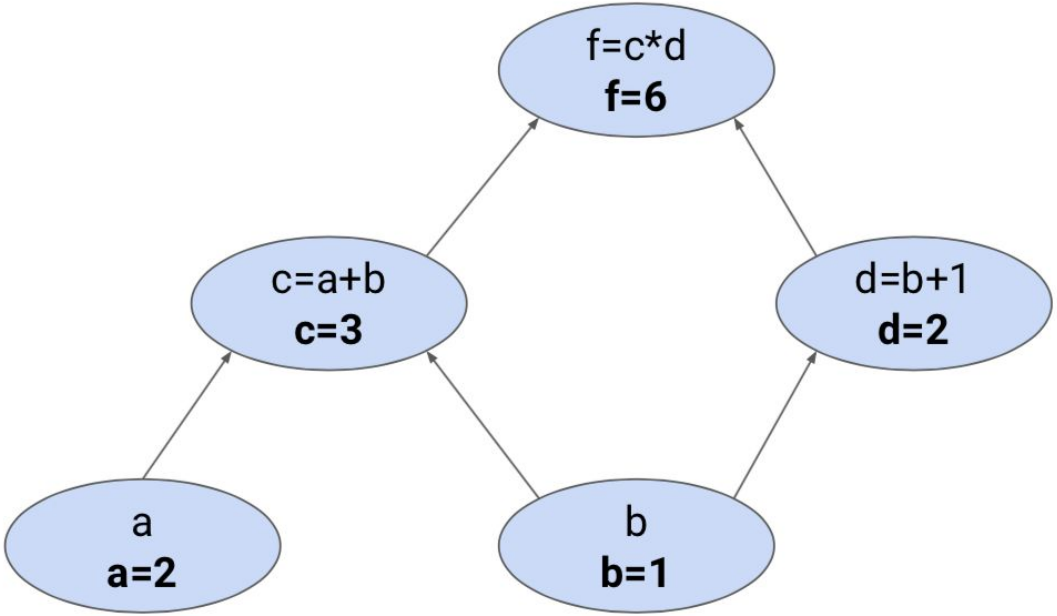
The gradient of the loss w.r.t. the weights.

$$\nabla_W L = \frac{\partial L}{\partial a}, \frac{\partial L}{\partial b}$$

For example, if we increase 'a' by a little bit, how does this affect L? (Does L increase, or decrease, and at what rate compared to our increase in 'a'?)

The gradient gives us the answer

$$\nabla_W L = \frac{\partial L}{\partial a}, \frac{\partial L}{\partial b}$$

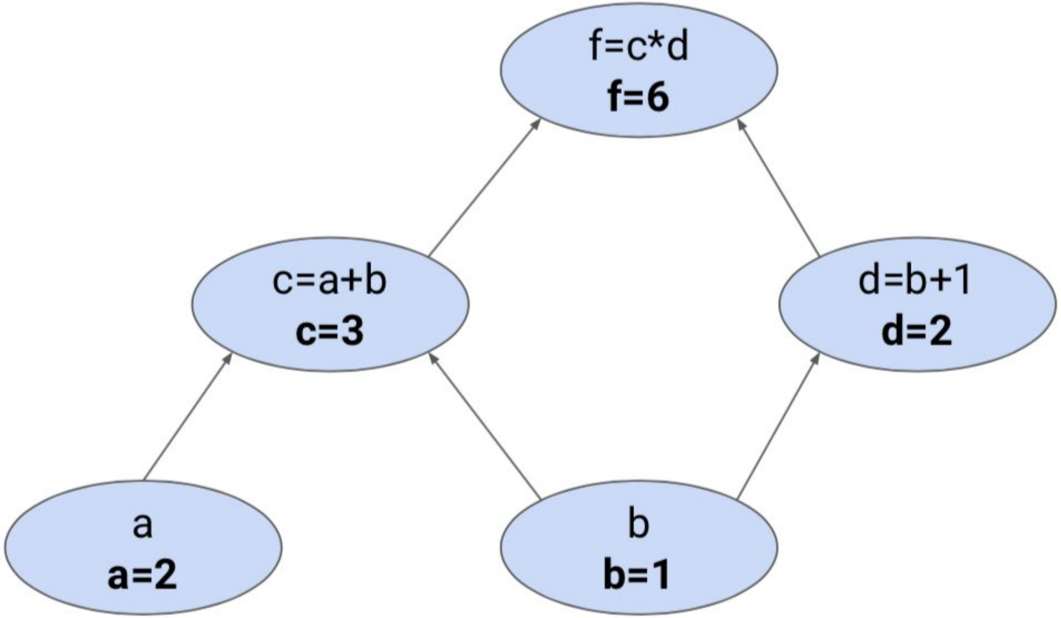


We got [2, 5].

If we increase a by epsilon, we expect f to increase by 2 * epsilon.

Likewise, if we increase b by epsilon, f should increase by 5 * epsilon.

Backprop: A method for efficiently computing gradients (by recursive application of the chain rule on a computational graph).

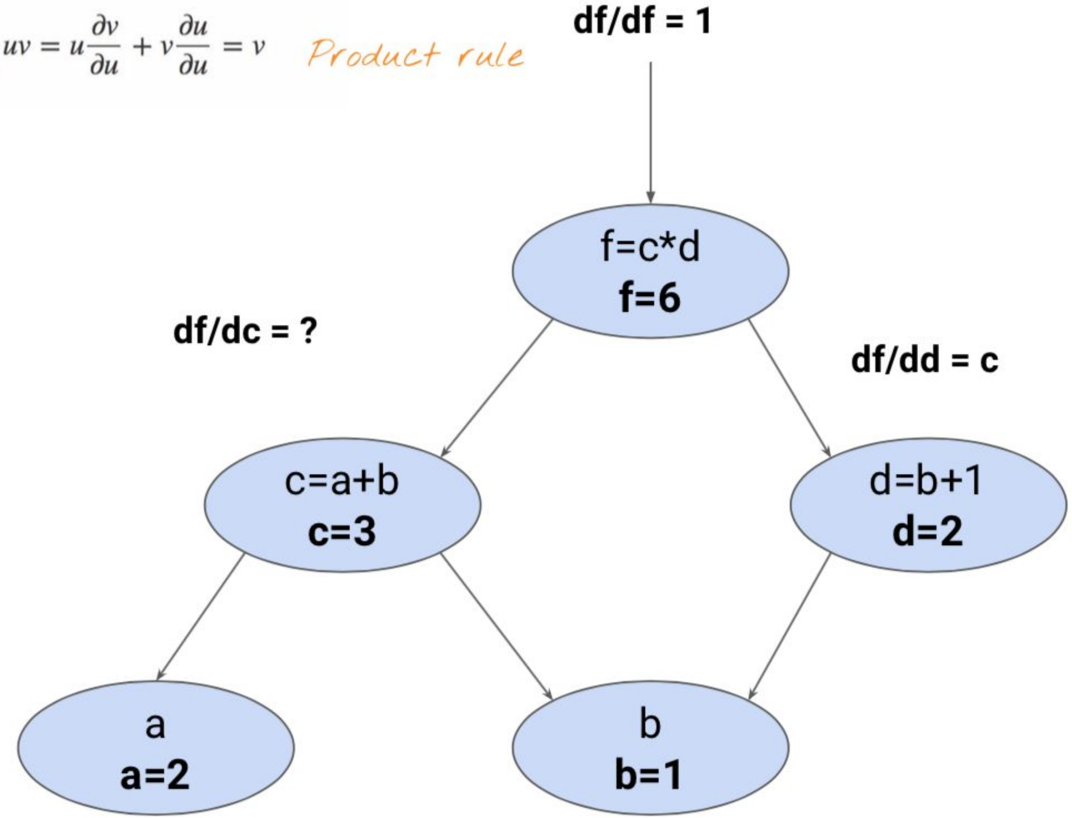


1) Compute the forward pass (finished here).

2) Starting from the output, begin propagating gradients backward along **edges in the graph.**

$$\frac{\partial}{\partial a}(a+b) = \frac{\partial a}{\partial a} + \frac{\partial b}{\partial a} = 1$$ Sum rule

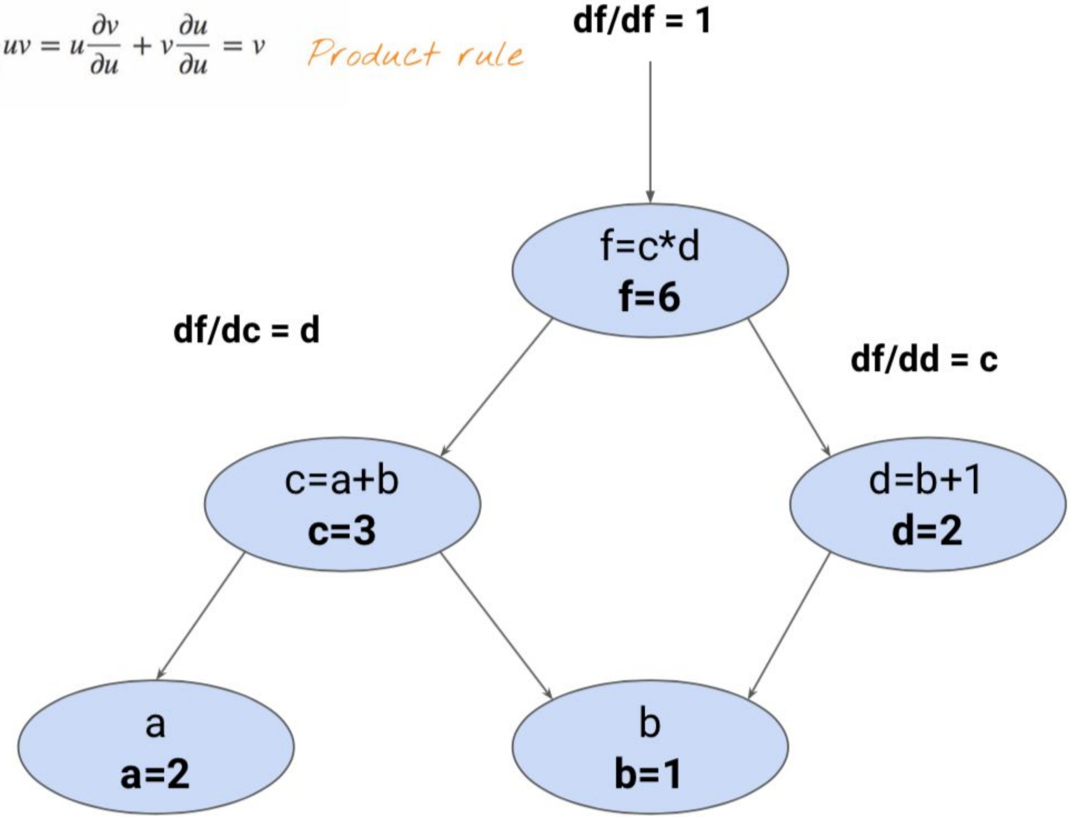
$$\frac{\partial}{\partial u}uv = u\frac{\partial v}{\partial u} + v\frac{\partial u}{\partial u} = v$$ Product rule



If we increase c by a little, then f increases at a rate of ?. So the gradient on this edge is ?.

$$\frac{\partial}{\partial a}(a+b) = \frac{\partial a}{\partial a} + \frac{\partial b}{\partial a} = 1$$ Sum rule

$$\frac{\partial}{\partial u}uv = u\frac{\partial v}{\partial u} + v\frac{\partial u}{\partial u} = v$$ Product rule

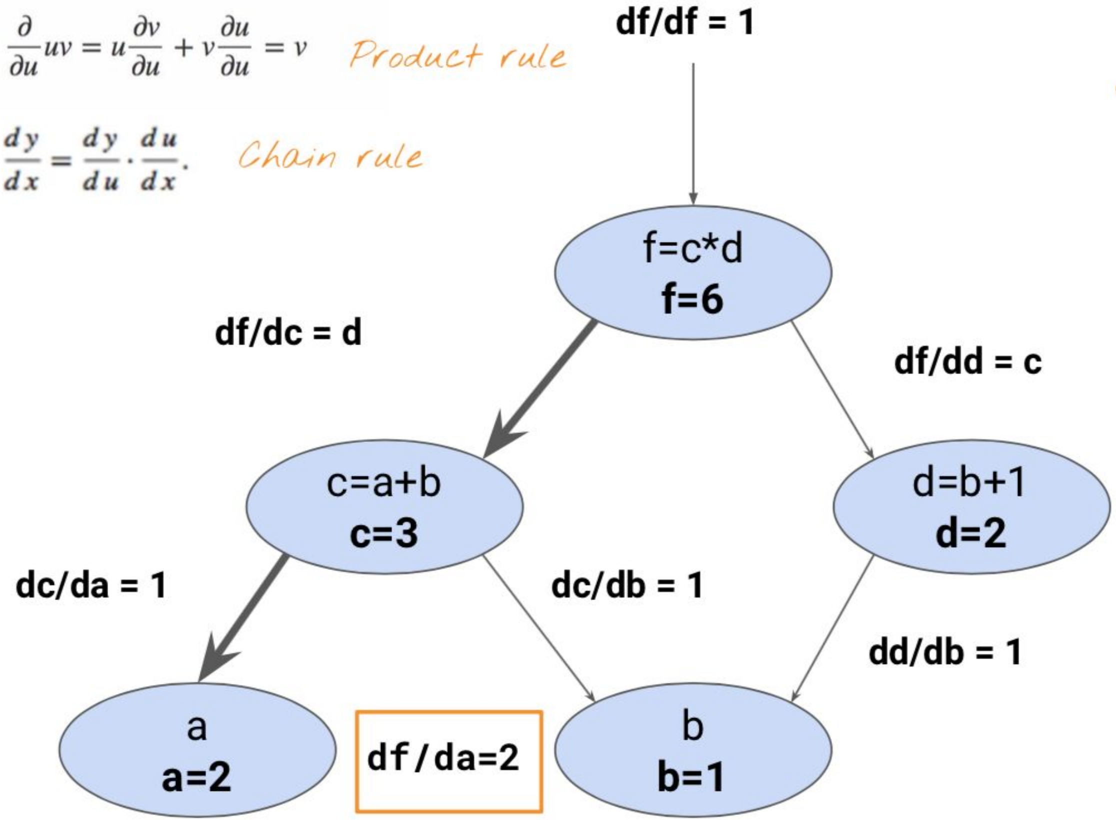


Product rule. Intuition: if we increase c by a little, then f increases at a rate of d. So the gradient on this edge is d.

$$\frac{\partial}{\partial a}(a+b) = \frac{\partial a}{\partial a} + \frac{\partial b}{\partial a} = 1$$ Sum rule

$$\frac{\partial}{\partial u}uv = u\frac{\partial v}{\partial u} + v\frac{\partial u}{\partial u} = v$$ Product rule

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}.$$ Chain rule



Now we can compute the gradient as the product along paths (another way of thinking about the chain rule!)

```
df/da = df/df * df/dc * dc/da
      = 1 * d * 1
      = 1 * 2 * 1
      = 2
```

Question: What does df/dc represent in the below diagram? Be specific if you can.

Answer:

1. The amount that f increases, if we increase c by a little :)
2. By product rule, the gradient is d. In this example, d=2.
3. In Backprop, a component in calculating the gradient of loss w.r.t. weights (df/da and df/db), using chain rule.