Computer Vision: Image Augumentation

Eng Teong Cheah

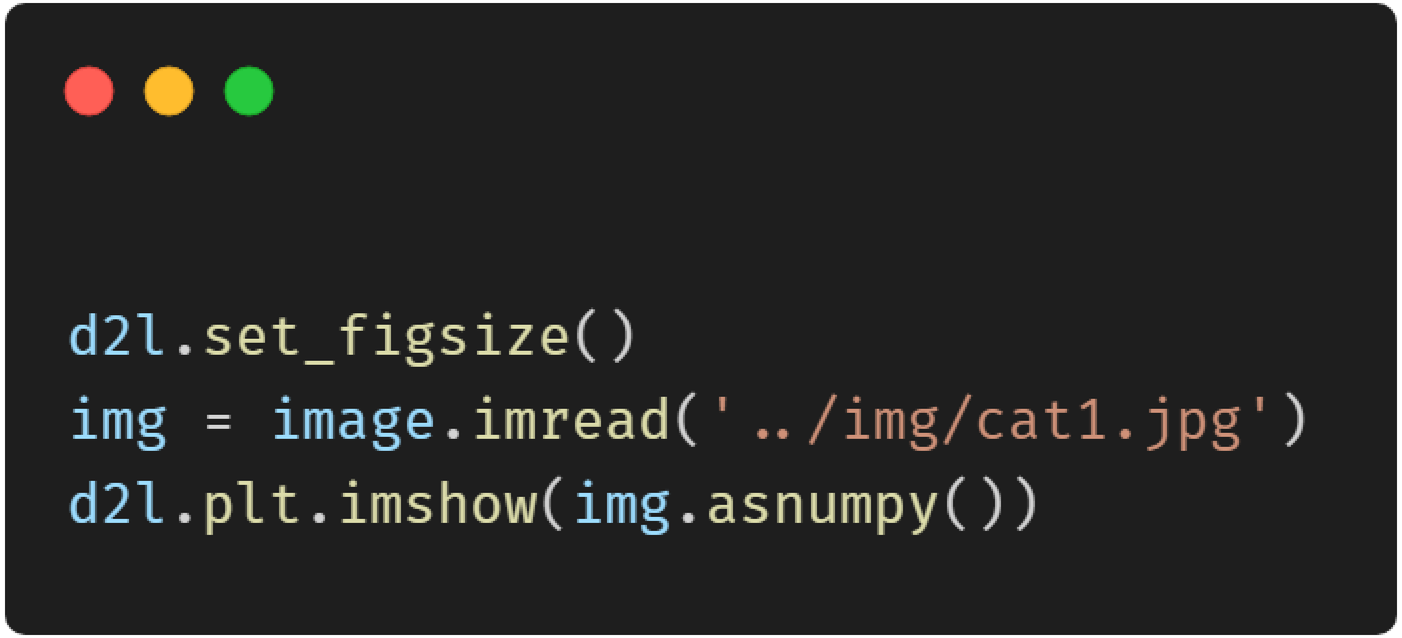
Contents

Image Augmentation

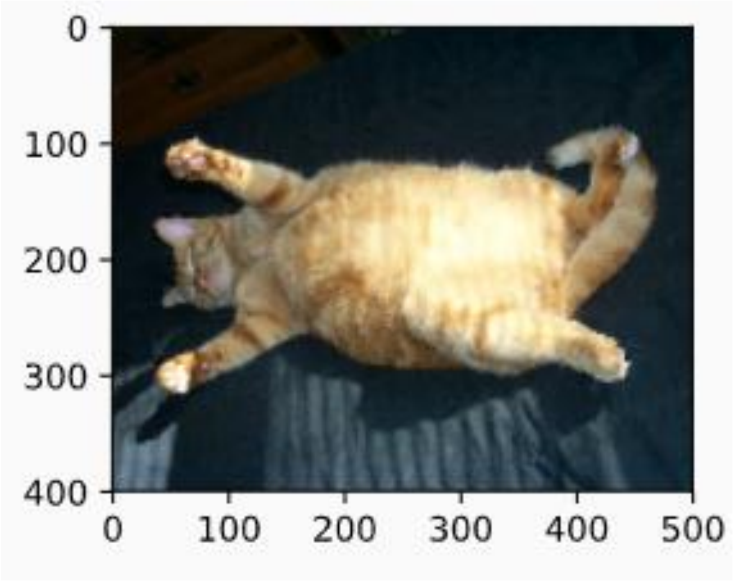
Image Augmentation

```
%matplotlib inline
import d2l
from mxnet import autograd, gluon, image, init, np,
npx
from mxnet.gluon import nn

npx.set_np()
```

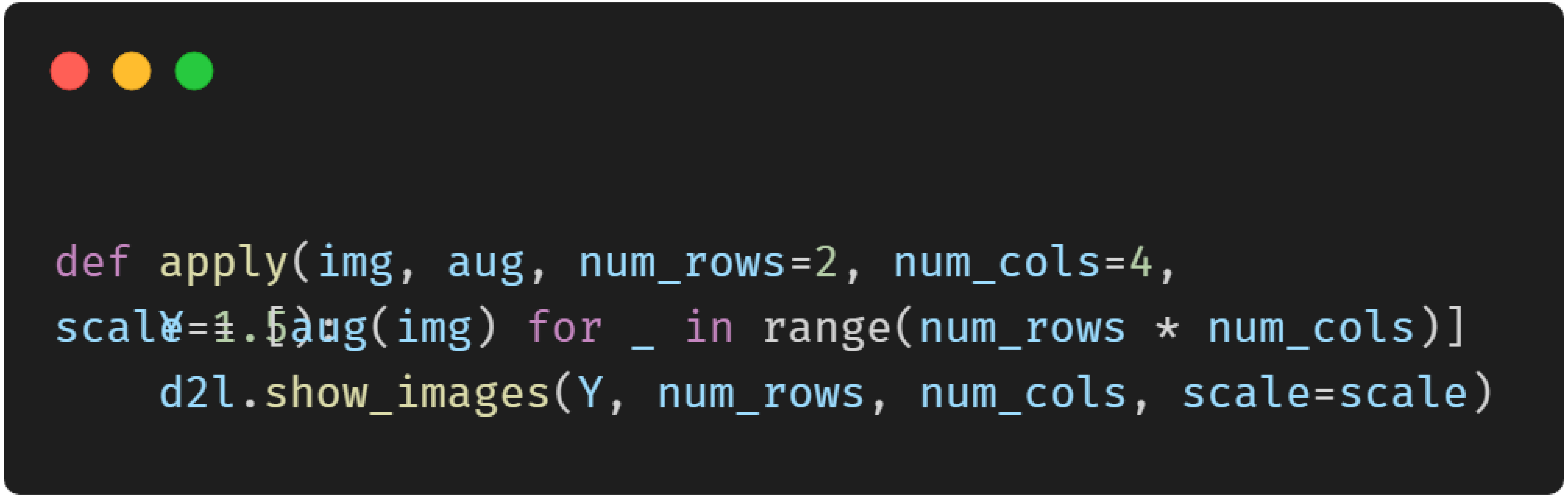
Common Image Augmentation Method



```
d2l.set_figsize()
img = image.imread('../img/cat1.jpg')
d2l.plt.imshow(img.asnumpy())
```

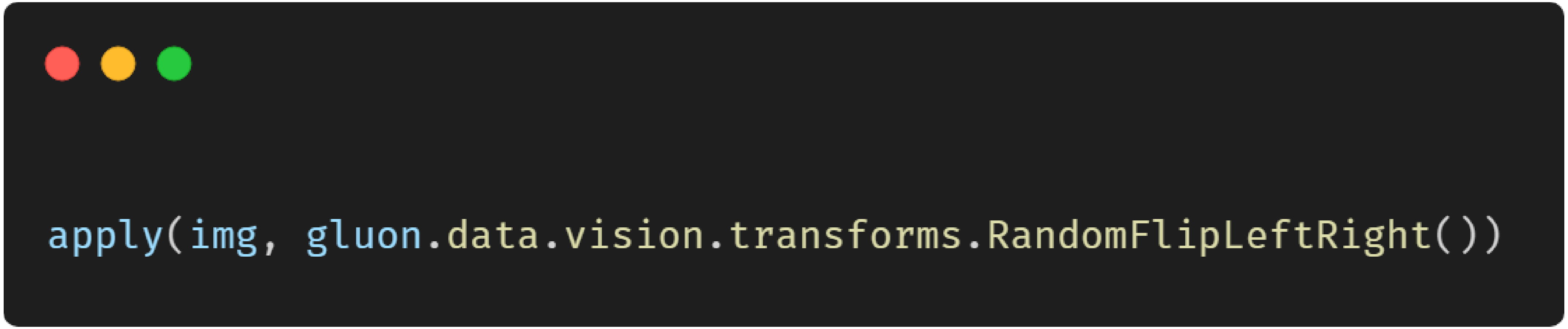


Common Image Augmentation Method

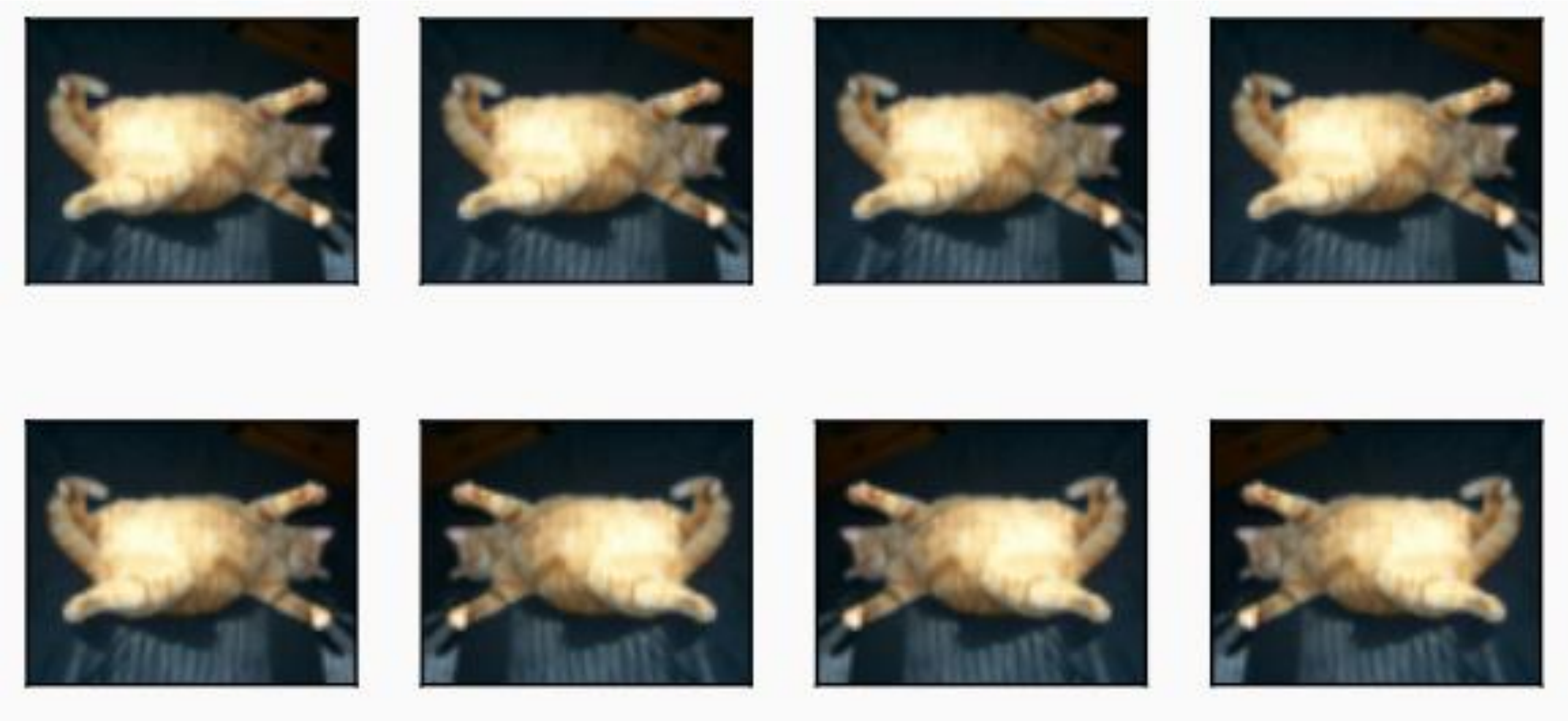


```
def apply(img, aug, num_rows=2, num_cols=4,
scale=1.5):
    Y = [aug(img) for _ in range(num_rows * num_cols)]
    d2l.show_images(Y, num_rows, num_cols, scale=scale)
```

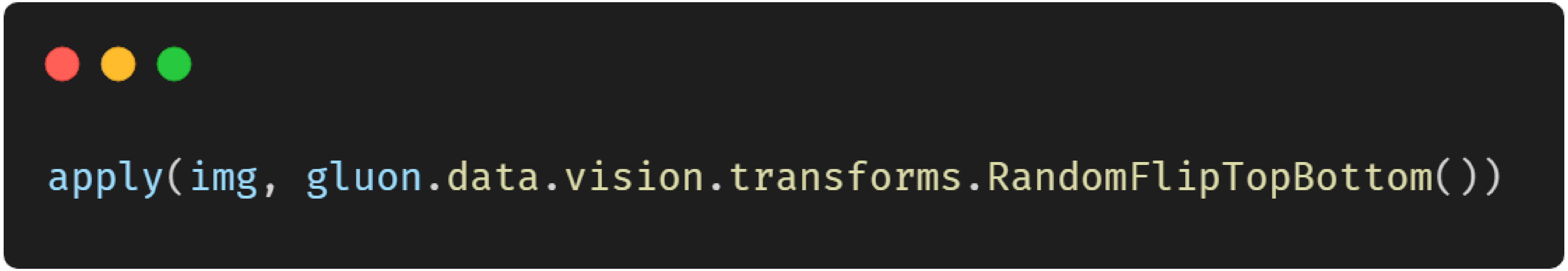
Flip and Crop



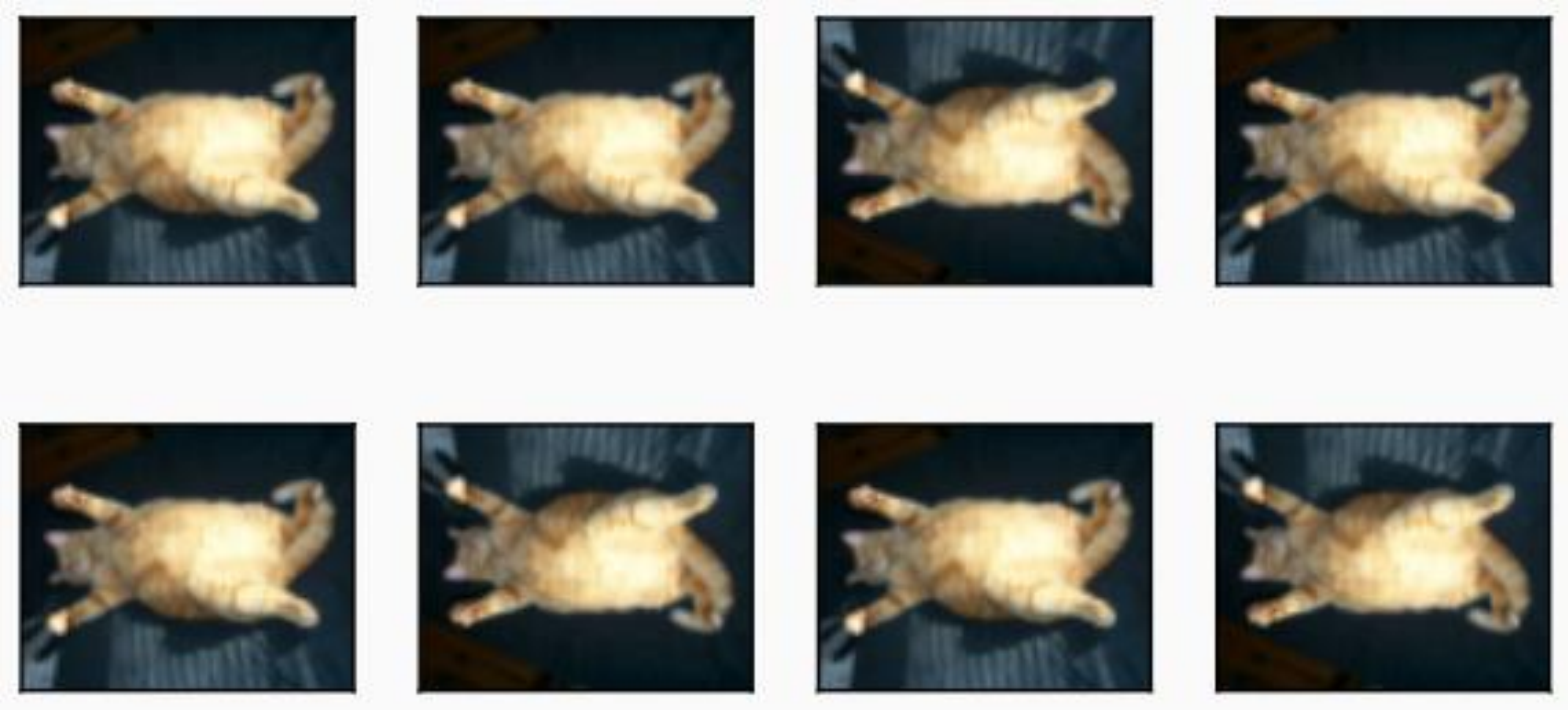
```
apply(img, gluon.data.vision.transforms.RandomFlipLeftRight())
```



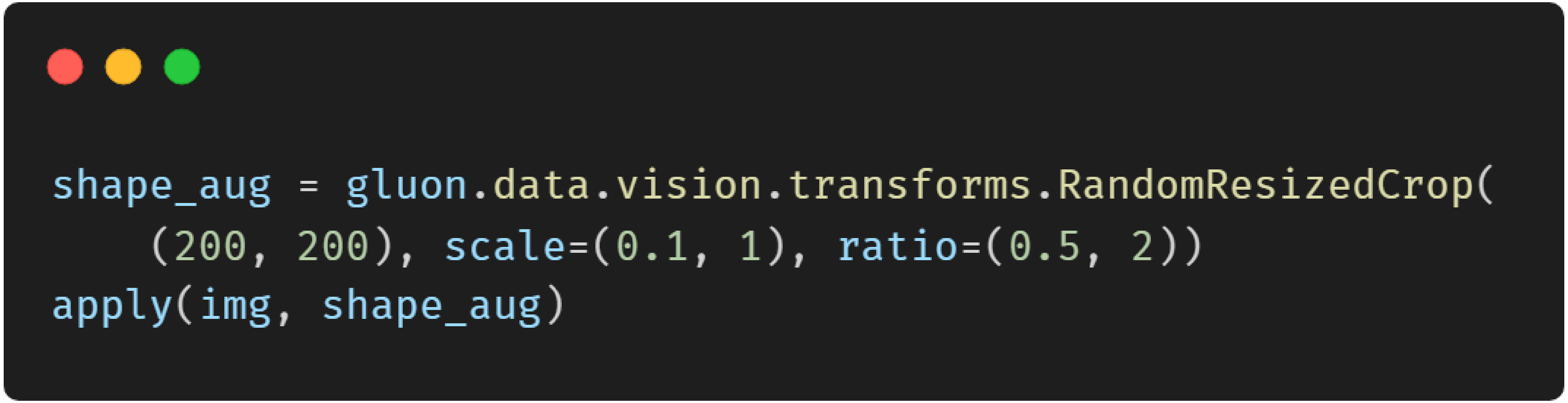
Flip and Crop



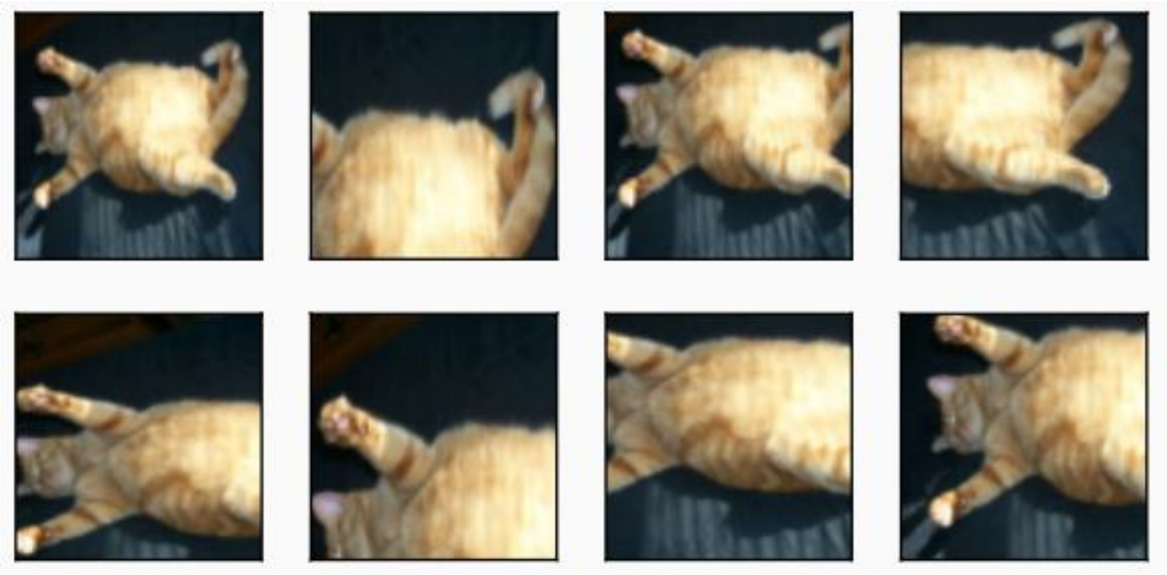
```
apply(img, gluon.data.vision.transforms.RandomFlipTopBottom())
```



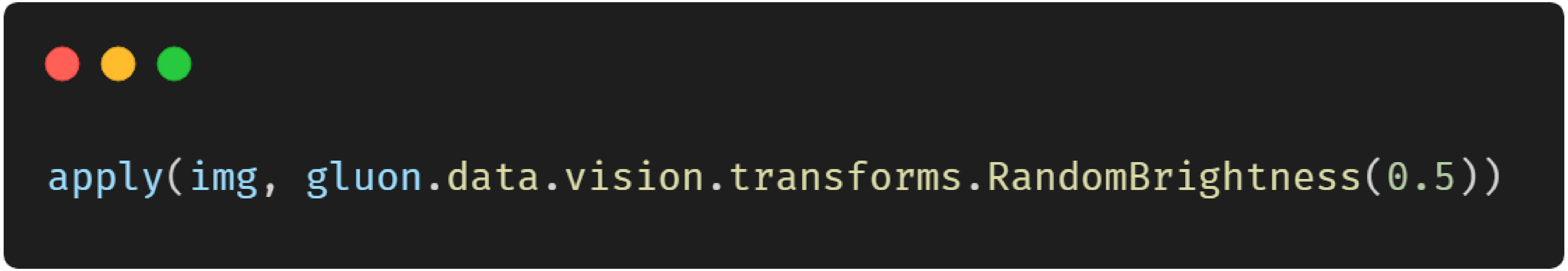
Flip and Crop



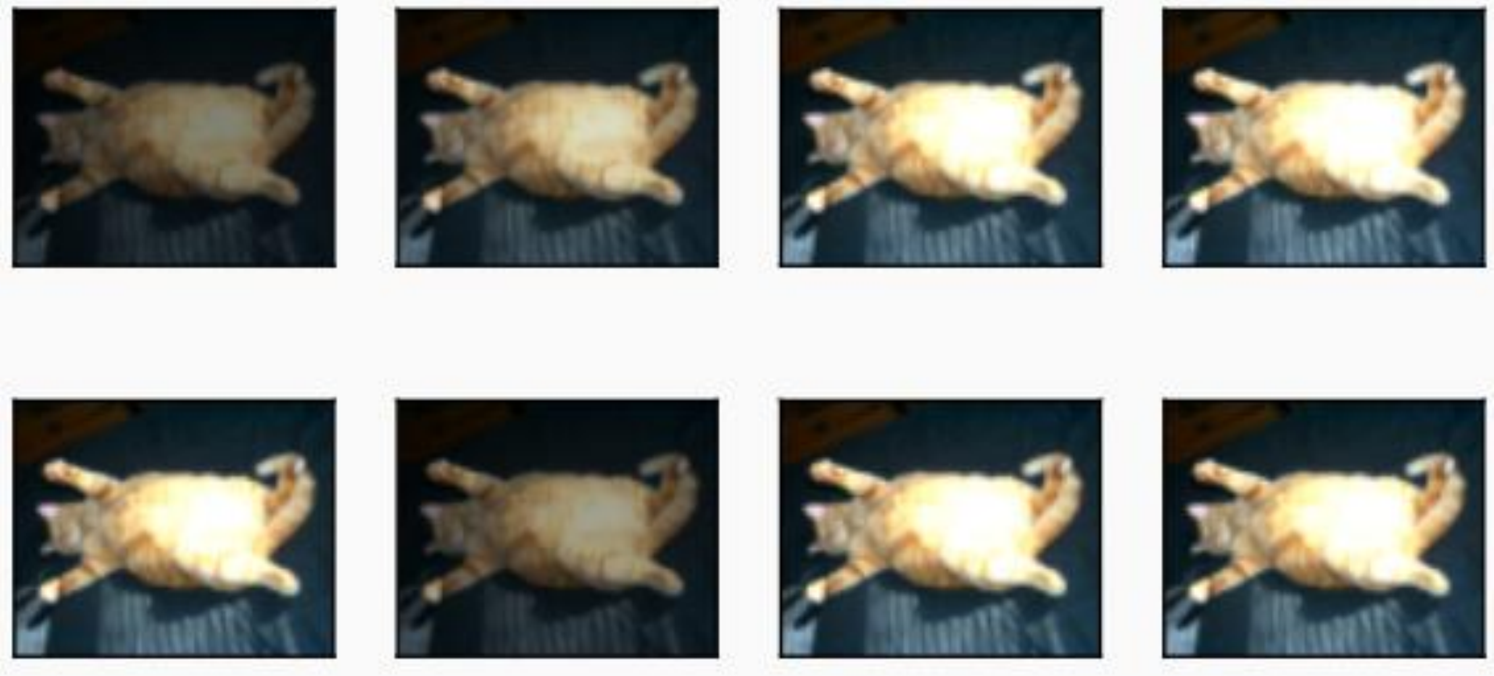
```
shape_aug = gluon.data.vision.transforms.RandomResizedCrop(
    (200, 200), scale=(0.1, 1), ratio=(0.5, 2))
apply(img, shape_aug)
```



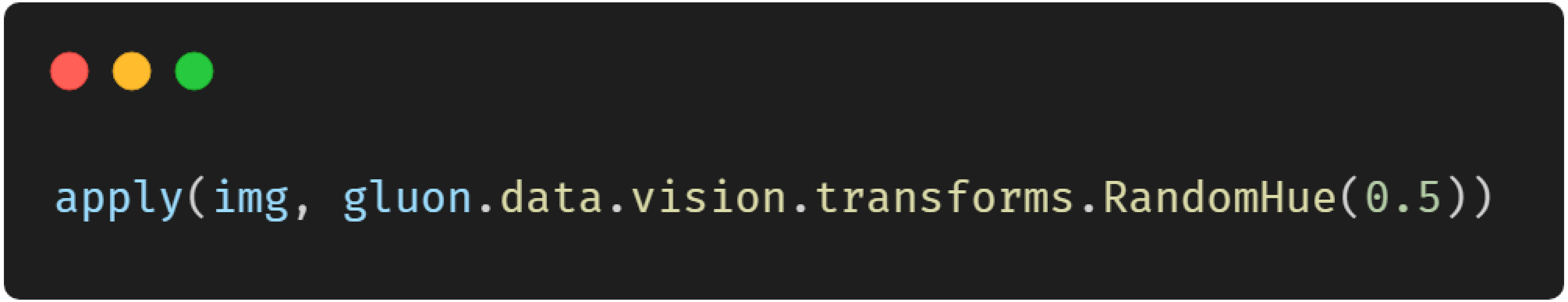
Change Color



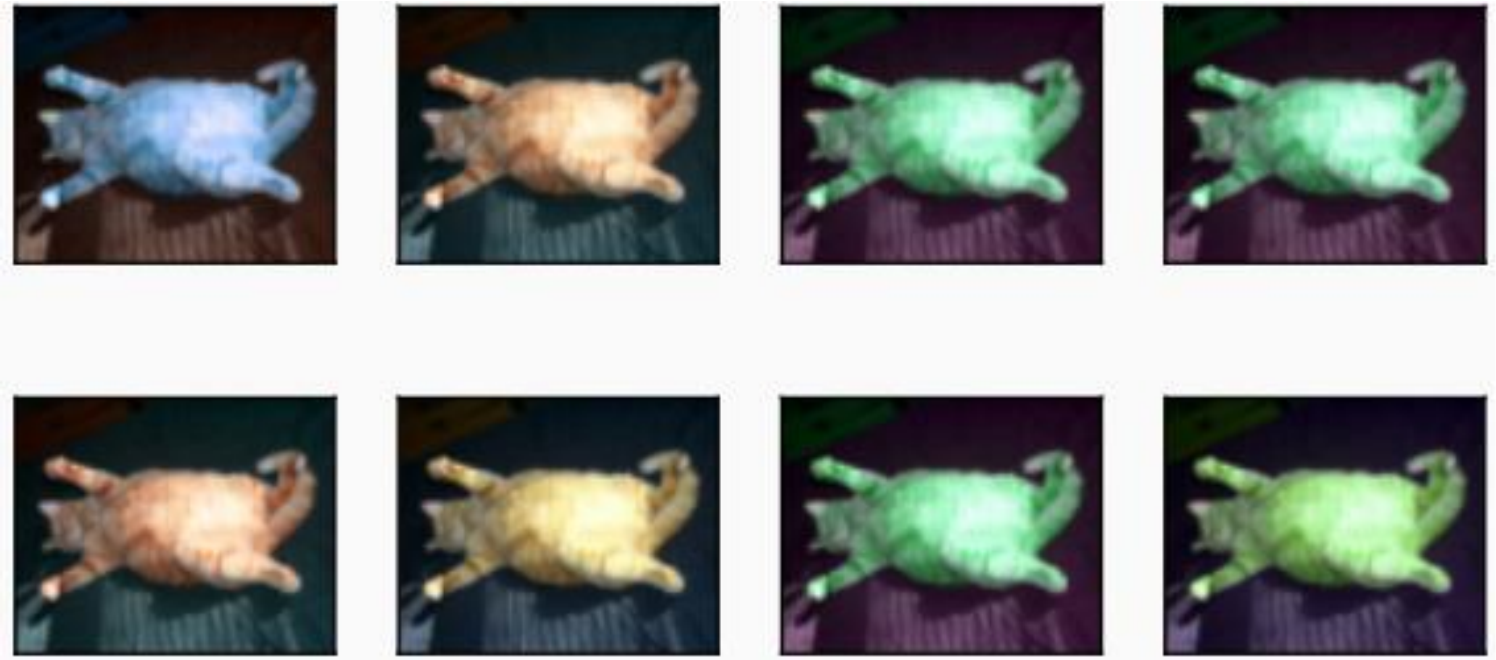
```
apply(img, gluon.data.vision.transforms.RandomBrightness(0.5))
```



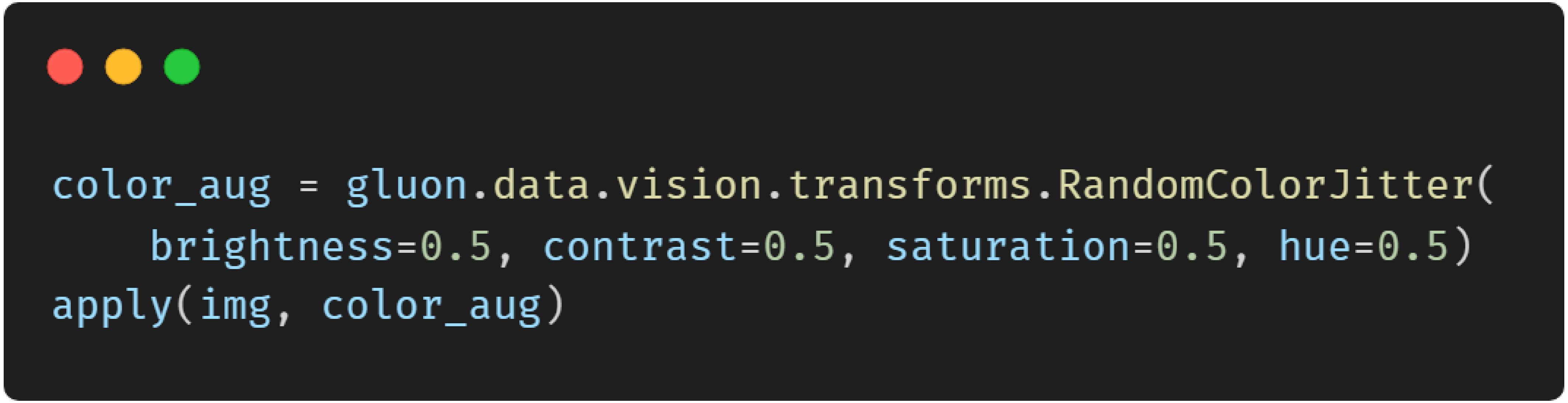
Change Color



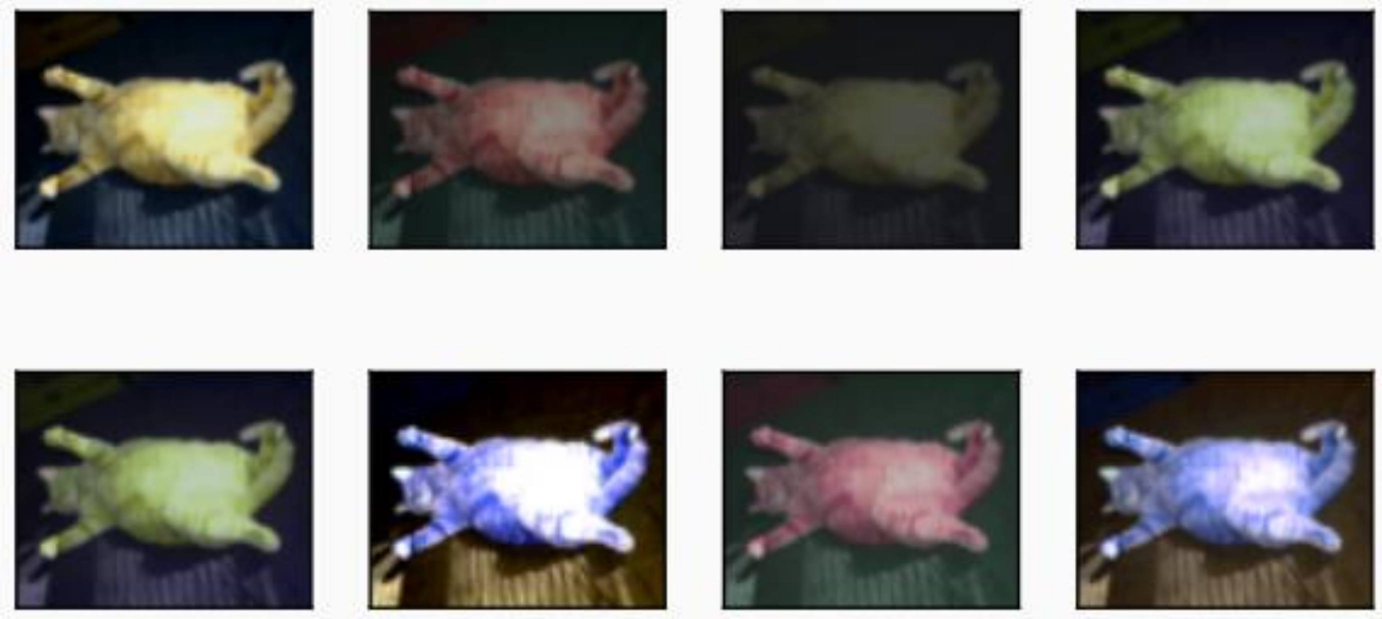
```
apply(img, gluon.data.vision.transforms.RandomHue(0.5))
```



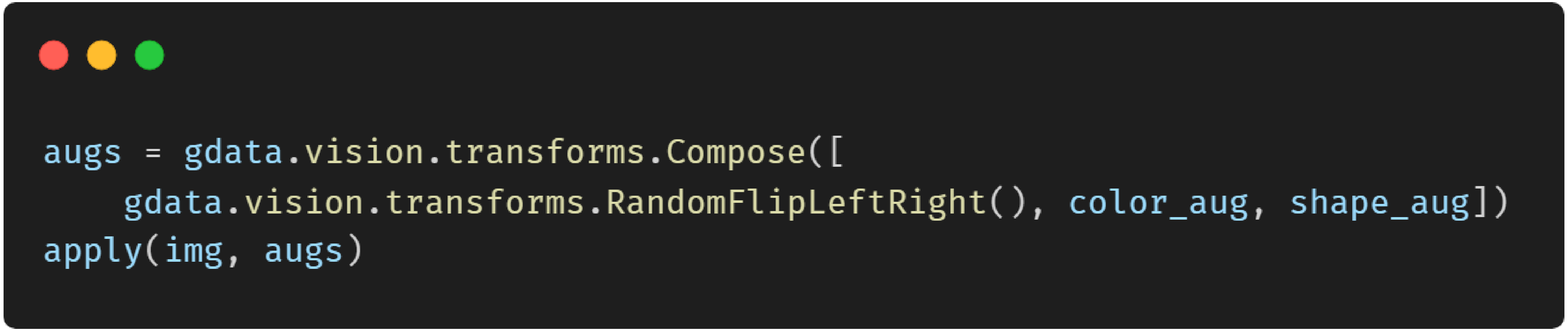
Change Color



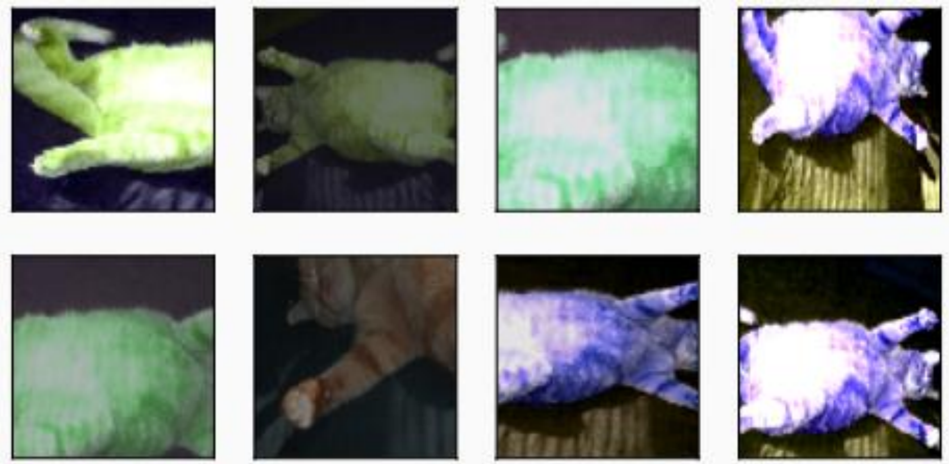
```
color_aug = gluon.data.vision.transforms.RandomColorJitter(
    brightness=0.5, contrast=0.5, saturation=0.5, hue=0.5)
apply(img, color_aug)
```



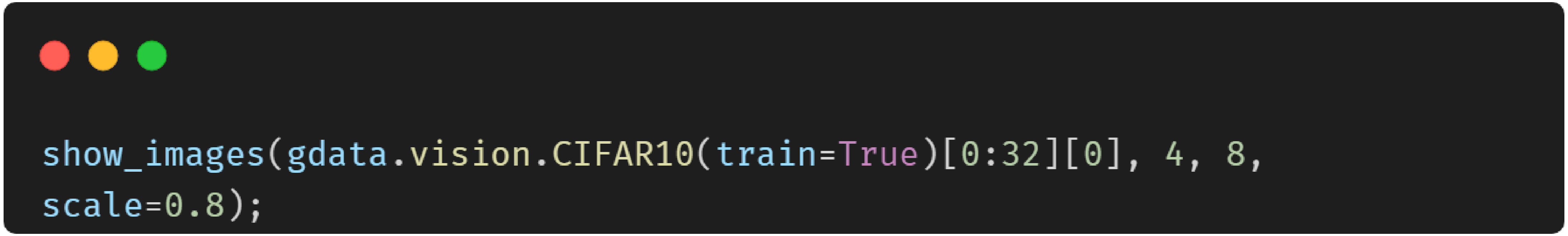
Overlying Multiple Image Augmentation Methods



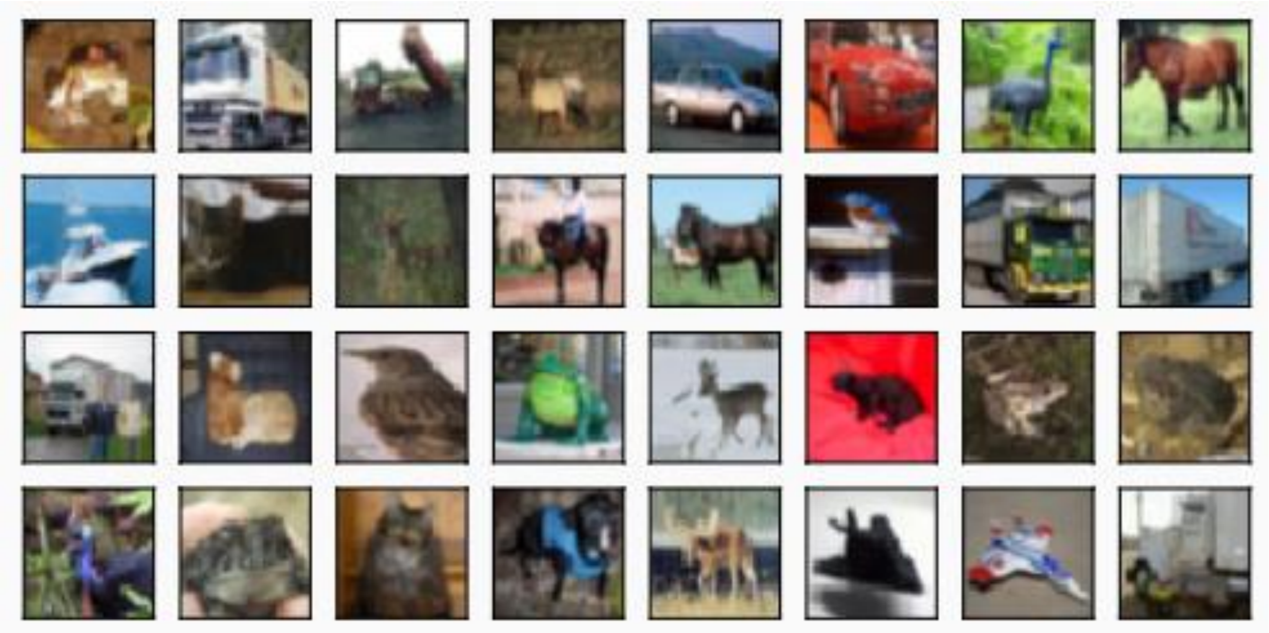
```
augs = gdata.vision.transforms.Compose([
    gdata.vision.transforms.RandomFlipLeftRight(), color_aug, shape_aug])
apply(img, augs)
```



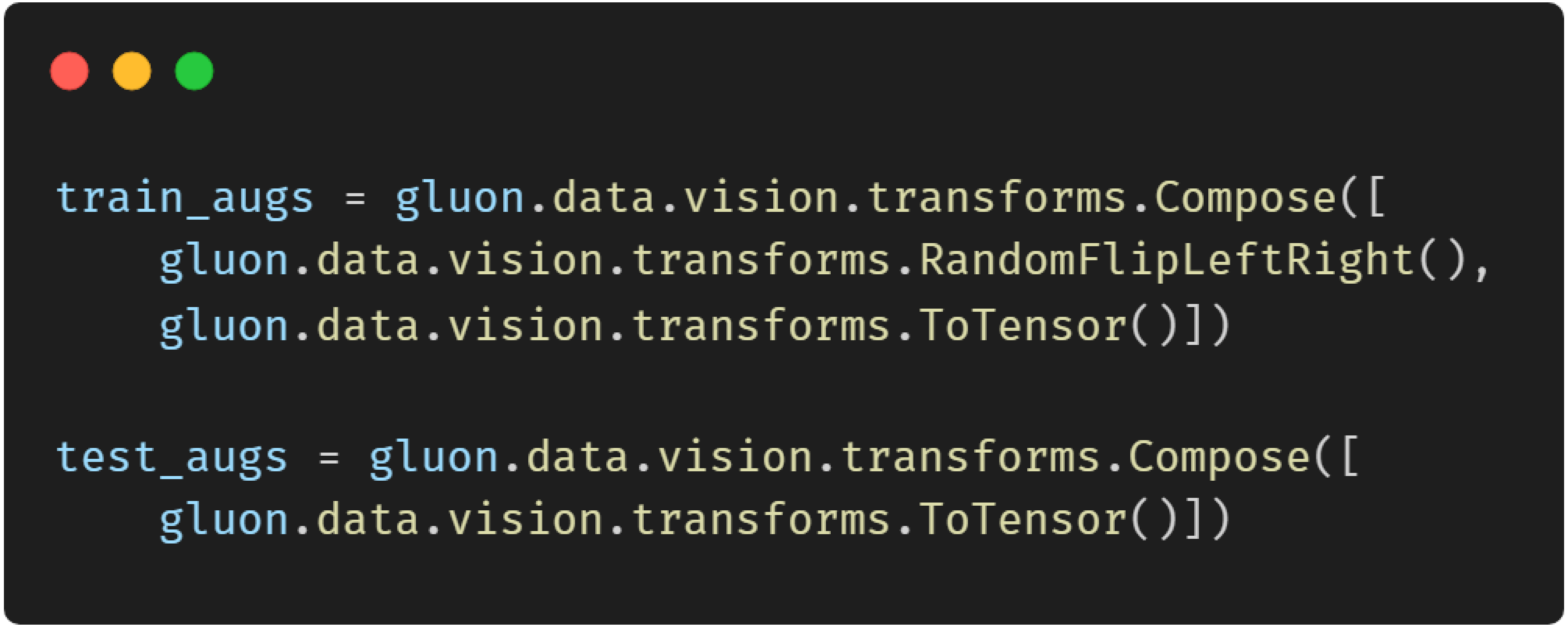
Using an Image Augmentation Training Model



```
show_images(gdata.vision.CIFAR10(train=True)[0:32][0], 4, 8,
scale=0.8);
```

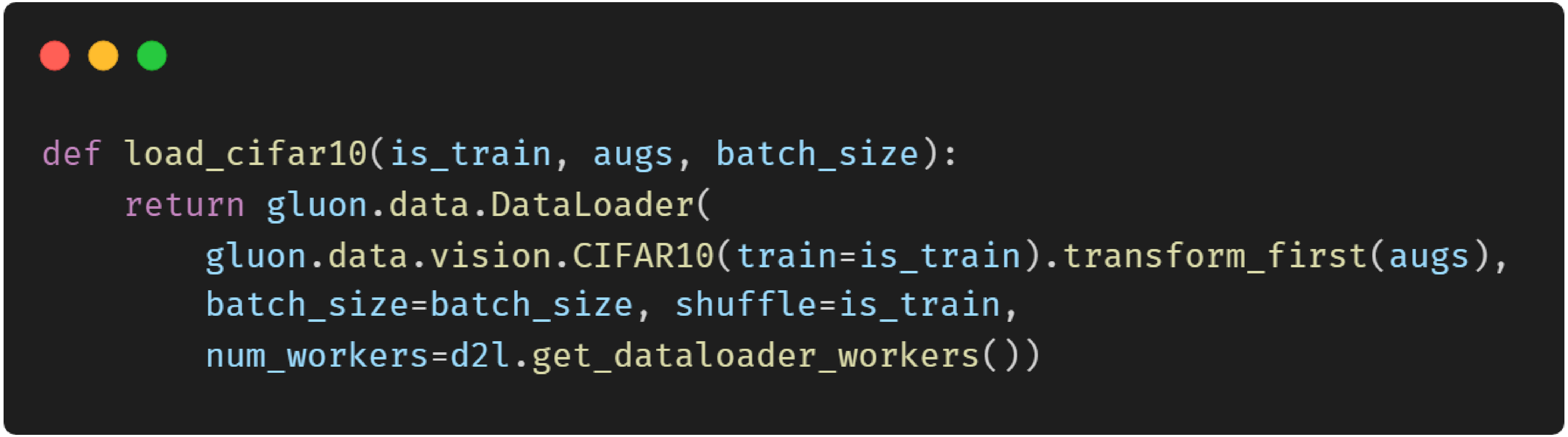


Using an Image Augmentation Training Model



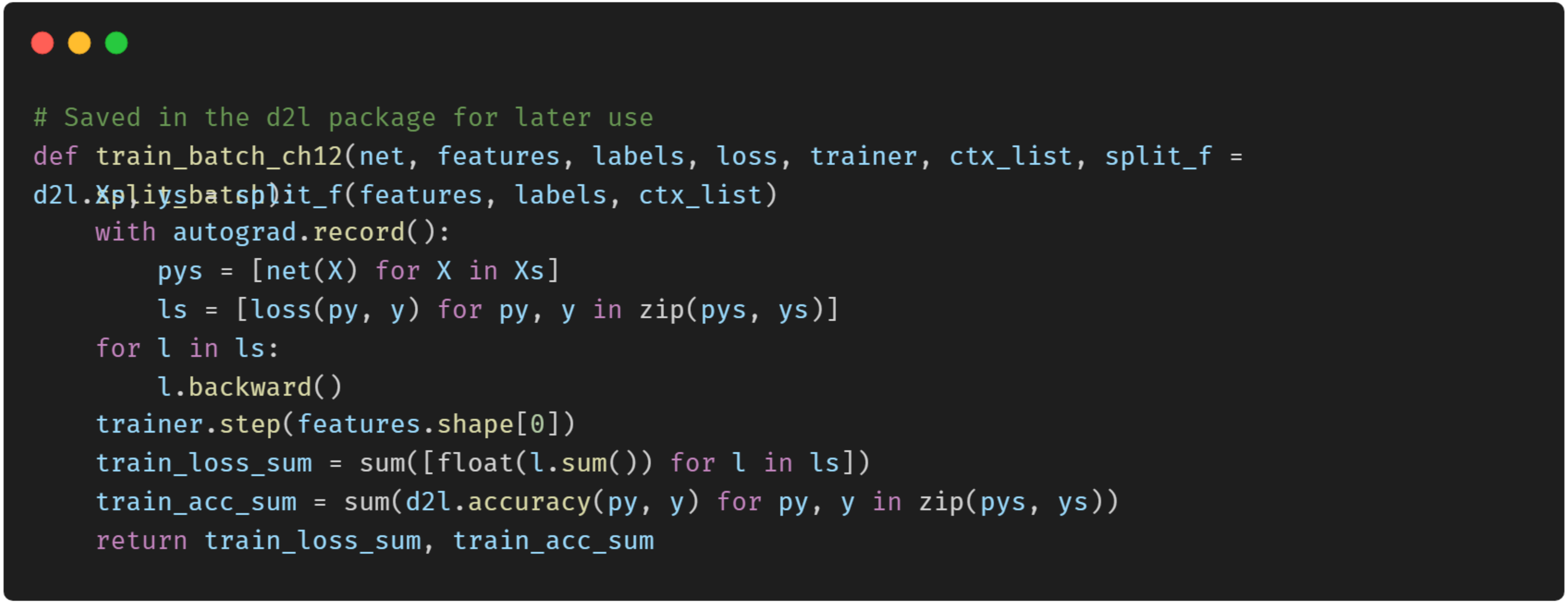
```
train_augs = gluon.data.vision.transforms.Compose([
    gluon.data.vision.transforms.RandomFlipLeftRight(),
    gluon.data.vision.transforms.ToTensor()])

test_augs = gluon.data.vision.transforms.Compose([
    gluon.data.vision.transforms.ToTensor()])
```

Using an Image Augmentation Training Model



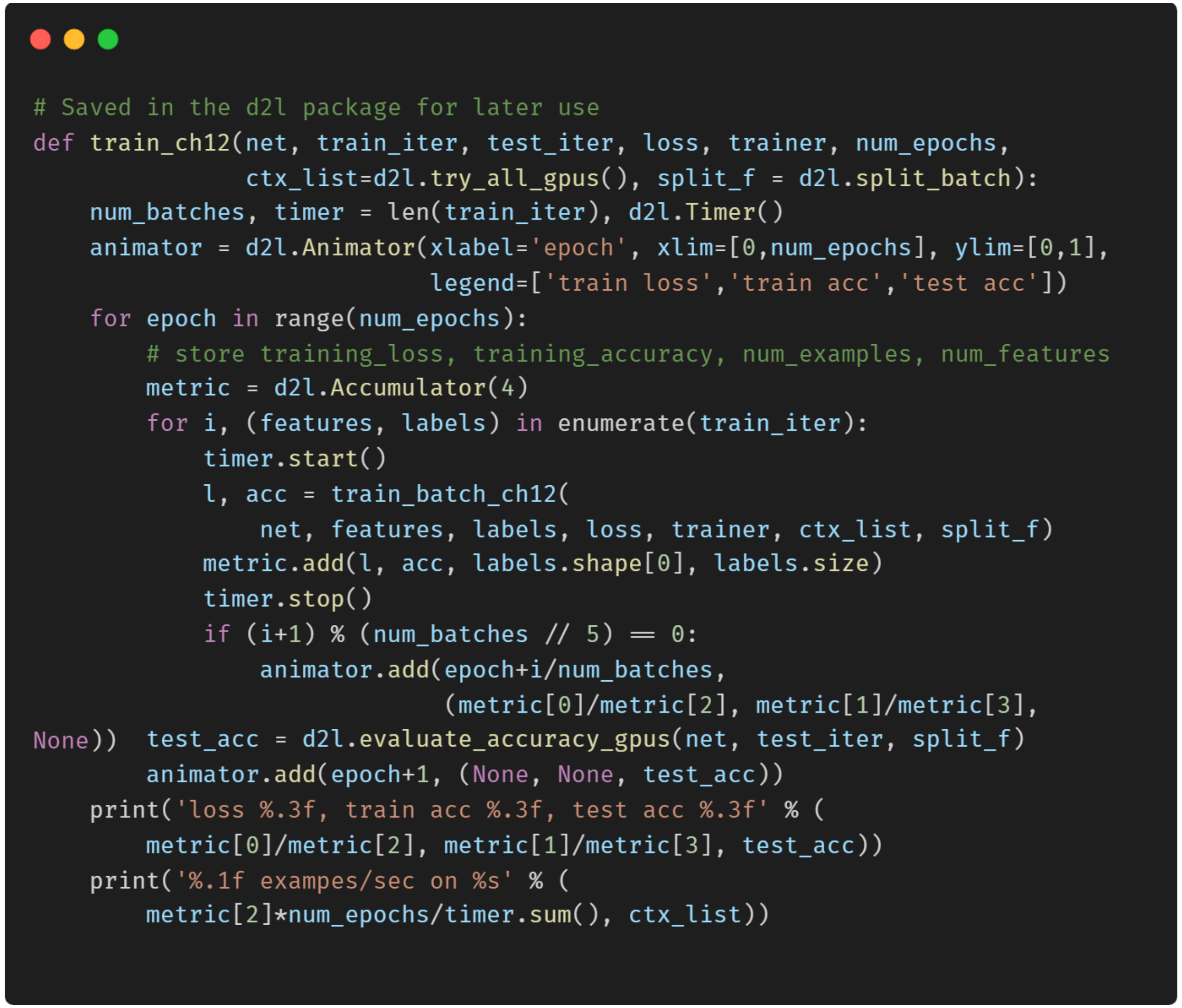
```python
def load_cifar10(is_train, augs, batch_size):
    return gluon.data.DataLoader(
        gluon.data.vision.CIFAR10(train=is_train).transform_first(augs),
        batch_size=batch_size, shuffle=is_train,
        num_workers=d2l.get_dataloader_workers())
```

Using a Multi-GPU Training Model



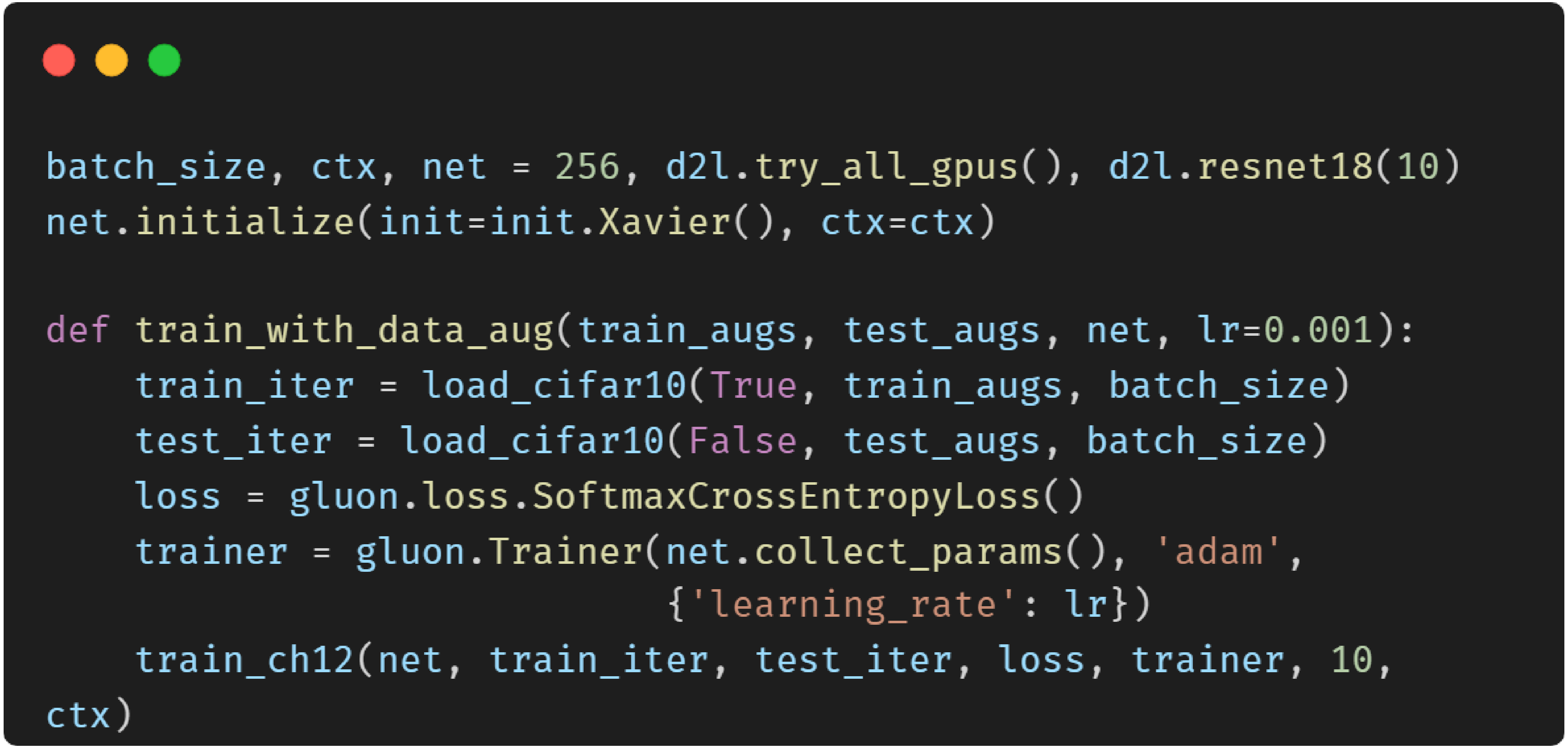
```
# Saved in the d2l package for later use
def train_batch_ch12(net, features, labels, loss, trainer, ctx_list, split_f =
d2l.split_batch):
    Xs, ys = split_f(features, labels, ctx_list)
    with autograd.record():
        pys = [net(X) for X in Xs]
        ls = [loss(py, y) for py, y in zip(pys, ys)]
    for l in ls:
        l.backward()
    trainer.step(features.shape[0])
    train_loss_sum = sum([float(l.sum()) for l in ls])
    train_acc_sum = sum(d2l.accuracy(py, y) for py, y in zip(pys, ys))
    return train_loss_sum, train_acc_sum
```

Using a Multi-GPU Training Model



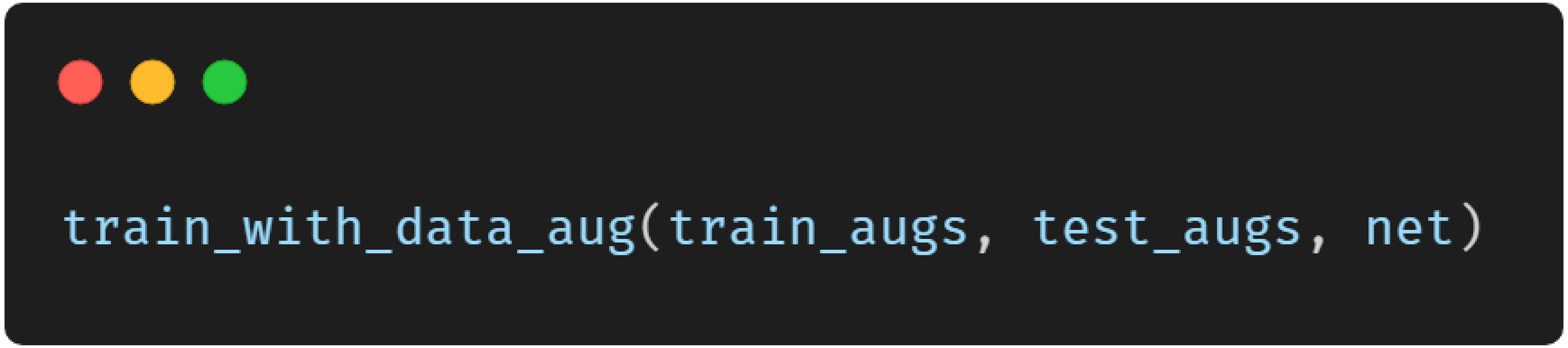
```
# Saved in the d2l package for later use
def train_ch12(net, train_iter, test_iter, loss, trainer, num_epochs,
               ctx_list=d2l.try_all_gpus(), split_f = d2l.split_batch):
    num_batches, timer = len(train_iter), d2l.Timer()
    animator = d2l.Animator(xlabel='epoch', xlim=[0,num_epochs], ylim=[0,1],
                            legend=['train loss','train acc','test acc'])
    for epoch in range(num_epochs):
        # store training_loss, training_accuracy, num_examples, num_features
        metric = d2l.Accumulator(4)
        for i, (features, labels) in enumerate(train_iter):
            timer.start()
            l, acc = train_batch_ch12(
                net, features, labels, loss, trainer, ctx_list, split_f)
            metric.add(l, acc, labels.shape[0], labels.size)
            timer.stop()
            if (i+1) % (num_batches // 5) == 0:
                animator.add(epoch+i/num_batches,
                             (metric[0]/metric[2], metric[1]/metric[3],
None))  test_acc = d2l.evaluate_accuracy_gpus(net, test_iter, split_f)
        animator.add(epoch+1, (None, None, test_acc))
    print('loss %.3f, train acc %.3f, test acc %.3f' % (
        metric[0]/metric[2], metric[1]/metric[3], test_acc))
    print('%.1f exampes/sec on %s' % (
        metric[2]*num_epochs/timer.sum(), ctx_list))
```

Using a Multi-GPU Training Model

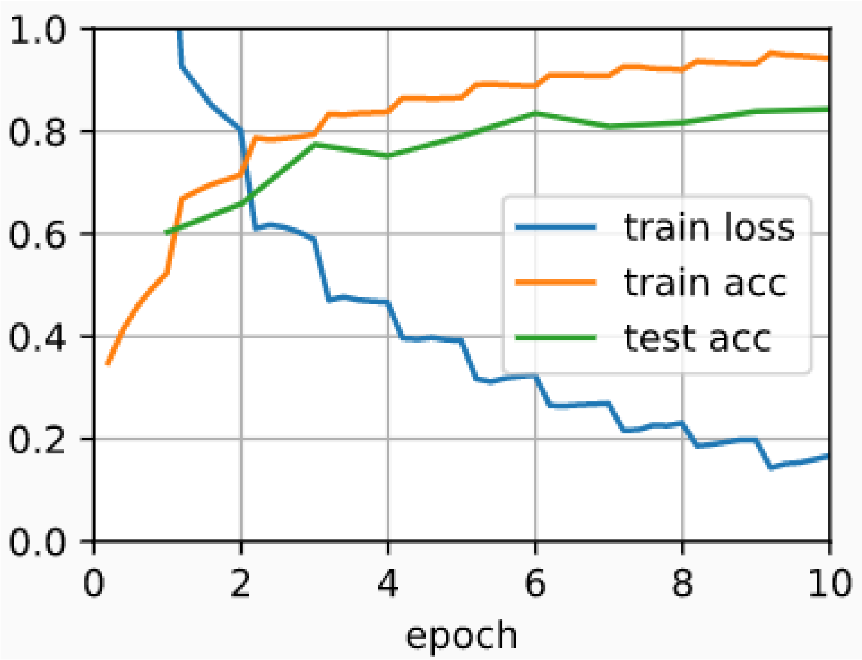


```
batch_size, ctx, net = 256, d2l.try_all_gpus(), d2l.resnet18(10)
net.initialize(init=init.Xavier(), ctx=ctx)

def train_with_data_aug(train_augs, test_augs, net, lr=0.001):
    train_iter = load_cifar10(True, train_augs, batch_size)
    test_iter = load_cifar10(False, test_augs, batch_size)
    loss = gluon.loss.SoftmaxCrossEntropyLoss()
    trainer = gluon.Trainer(net.collect_params(), 'adam',
                            {'learning_rate': lr})
    train_ch12(net, train_iter, test_iter, loss, trainer, 10,
ctx)
```

Using a Multi-GPU Training Model



```
train_with_data_aug(train_augs, test_augs, net)
```



Thank You !

Does anyone have any questions?

Twitter: @walkercet
Blog: https://ceteongvanness.wordpress.com

Resources

Dive into Deep Learning