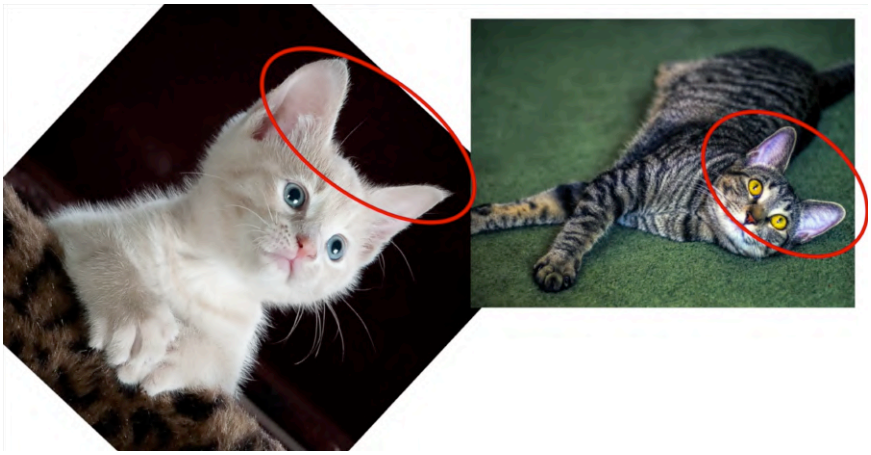
Image Augmentation

You'll be looking a lot at Image Augmentation this week.

Image Augmentation is a very simple, but very powerful tool to help you avoid overfitting your data. The concept is very simple though: If you have limited data, then the chances of you having data to match potential future predictions is also limited, and logically, the less data you have, the less chance you have of getting accurate predictions for data that your model hasn't yet seen. To put it simply, if you are training a model to spot cats, and your model has never seen what a cat looks like when lying down, it might not recognize that in future.

Augmentation simply amends your images on-the-fly while training using transforms like rotation. So, it could 'simulate' an image of a cat lying down by rotating a 'standing' cat by 90 degrees. As such you get a cheap way of extending your dataset beyond what you have already.

To learn more about Augmentation, and the available transforms, check out https://github.com/keras-team/keras-preprocessing -- and note that it's referred to as *pre*processing for a very powerful reason: that it doesn't require you to edit your raw images, nor does it amend them for you on-disk. It does it in-memory as it's performing the training, allowing you to experiment without impacting your dataset.



```
train_datagen = ImageDataGenerator(rescale=1./255)
```

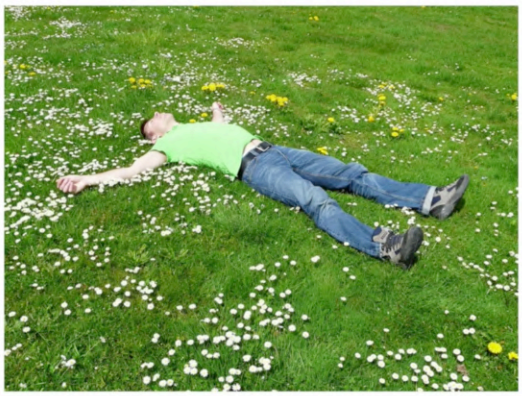
```
# Updated to do image augmentation
train_datagen = ImageDataGenerator(
      rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')
```

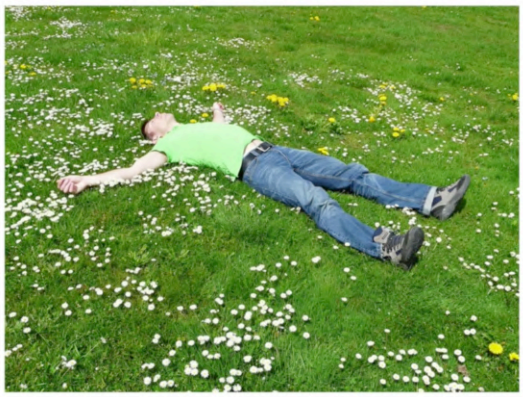
```
# Updated to do image augmentation
train_datagen = ImageDataGenerator(
      rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')
```

Rotation range is a range from 0-180 degrees with which to randomly rotate images. So in this case, the image will rotate by random amount between 0 and 40 degrees.

```
# Updated to do image augmentation
train_datagen = ImageDataGenerator(
      rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')
```

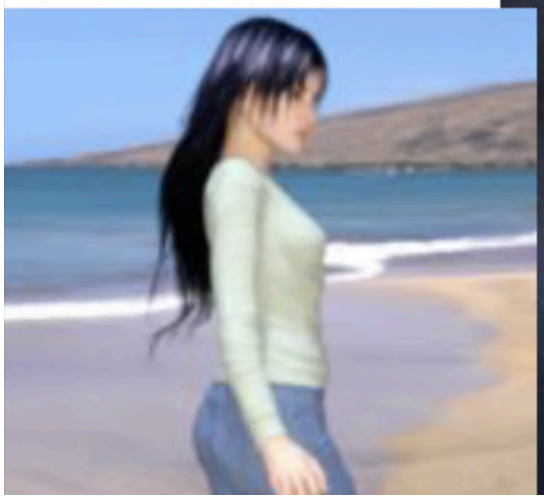
Shifting, moves the image around inside its frame. Many pictures have the subject centered. So if we train based on those kind of images, we might over-fit for that scenario. These parameters specify, as a proportion of the image size, how much we should randomly move the subject around. So in this case, we might offset it by 20 percent vertically or horizontally.





```
# Updated to do image augmentation
train_datagen = ImageDataGenerator(
      rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')
```

Shearing is also quite powerful. So for example, consider the image on the right. We know that it's a person. But in our training set, we don't have any images of a person in that orientation. However, we do have an image like this one, where the person is oriented similarly. So if we shear that person by skewing along the x-axis, we'll end up in a similar pose. That's what the shear_range parameter gives us. It will shear the image by random amounts up to the specified portion in the image. So in this case, it will shear up to 20 percent of the image.

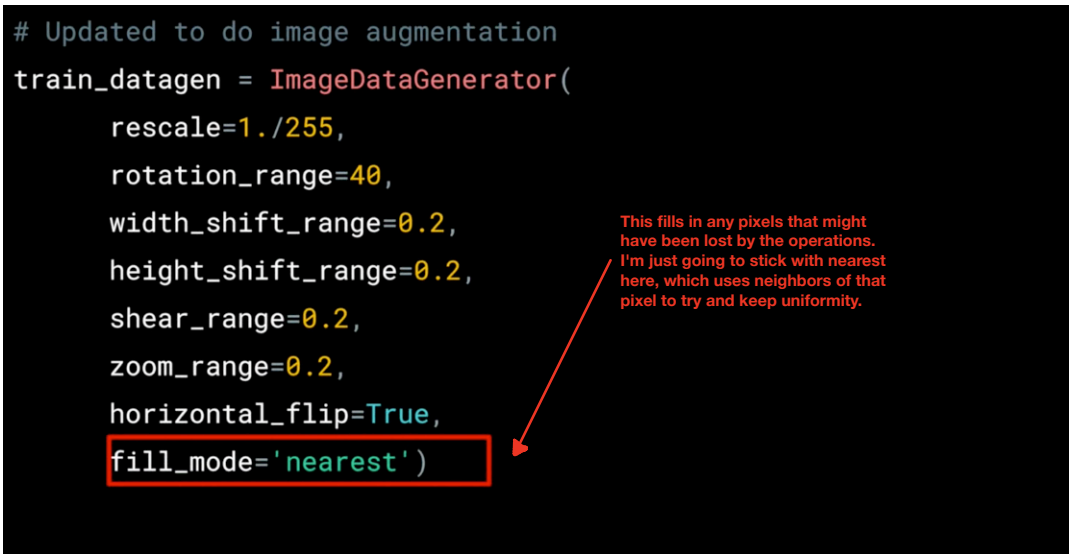


```
# Updated to do image augmentation
train_datagen = ImageDataGenerator(
      rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')
```

Zoom can also be very effective. For example, consider the image on the right. It's obviously a woman facing to the right. Our image on the left is from the humans or horses training set. It's very similar but it zoomed out to see the full person. If we zoom in on the training image, we could end up with a very similar image to the one on the right. Thus, if we zoom while training, we could spot more generalized examples like this one. So you zoom with code like this. The 0.2 is a relative portion of the image you will zoom in on. So in this case, zooms will be a random amount up to 20 percent of the size of the image.

```
# Updated to do image augmentation
train_datagen = ImageDataGenerator(
      rescale=1./255,
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')
```

Another useful tool is horizontal flipping. So for example, if you consider the picture on the right, we might not be able to classify it correctly as our training data doesn't have the image of a woman with her left hand raised, it does have the image on the left, where the subjects right arm is raised. So if the image were flipped horizontally, then it becomes more structurally similar to the image on the right and we might not over-fit to right arm raisers. To turn on random horizontal flipping, you just say horizontal_flip equals true and the images will be flipped at random.



What have we seen so far?

This week you looked at the really useful tool that TensorFlow gives you with Image Augmentation. With it, you can effectively simulate a larger dataset from a smaller one with tools to move images around the frame, skew them, rotate them, and more. It can be an effective tool in fixing overfitting. Hopefully you found it useful!