An epoch: A full cycle through the whole data set
batch size: number of samples to work through before updating model parameters.
Stochastic gradient descent means batch size is 1.
batch gradient descent means the batch size is the size of the training set (the whole thing)
mini-batch gradient descent is some batch size between SGD and BGD (i.e. 1 < batch size < size of training set)
learning rate annealing: decreasing the learning rate as you learn
cycle-len = n (from SGD R) means In SGD R for each cosine annealing phase, you change the learning rate every mini batch, and you reset it (i.e. make hte learning rate really high) every n epochs
cycle-mult = n means multiply the length of the cycle by n each time
learn.unfreeze() unfreezes all the layers so all the layers of the model are subject to parameter updates from training
differential learning rates: an array of 3 learning rates, the last one only affects the layers you added, the middle represents the middle conv layers, and the first learning rate affects the first (basic corners, edges, textures, etc) layers.
test time augmentations: for a sample in the test set, augment it a few times, and then take the average inference for the sample and the augmented samples.
data augmentation in general is only during training time
padding in general isn't helpful, but zooming can be
Sliding window for image detection: not great for training, good for test time augmentation
Tip: start training on small images, and then increase the size.
If you unfreeze the layers, do an epoch or two, and the losses don't change, it means your data is very similar to imagenet's data (or whatever your pretrained model was trained on)
***Since architectures are fixed, how does the first layer deal with different sized images? do you just realize everything to fit it in there?
***precompute = false vs true?
*** Why does a NN over fit when you train it too long? From a loss function persective, is it just more likely to find a steep local minimum?
Other Notes:
Review: easy steps to train a world class image classifier:

1. Enable data augmentation and precompute = True
2. use lr-find() to find the highest learning rate st loss is still clearly improving
3. train last layer from precomputed activations for 1 to 2 epochs
4. Train last layer with data augmentation (i.e. precompute = False) for 2 to 3 epochs with cycle-len = 1
5. Unfreeze all layers
6. set earlier layers to 3x - 10x lower learning rate than next higher layer
7. use lr-find() again
8. train full network with cycle-mult = 2 until over-fitting