WEEK – 2 (Augmentation)

data generator is

how your images get automatically labeled based on the directories that they're in.

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.

**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.**

WEEK – 3 (Transfer Learning)

**we were able to take somebody else's model,**

**a massive model that took**

**a long time with a lot of data to train from,**

**but you were able to build on that**

**to make your models even better.**

Most of these models are a series of convolutional layers followed by one or a few dense (or fully connected) layers.

Include\_top lets you select if you want the final dense layers or not.

* the convolutional layers work as feature extractors. They identify a series of patterns in the image, and each layer can identify more elaborate patterns by seeing patterns of patterns.
* the dense layers are capable of interpreting the found patterns in order to classify: this image contains cats, dogs, cars, etc.

About the weights:

* the weights in a convolutional layer are fixed-size. They are the size of the kernel x filters. Example: a 3x3 kernel of 10 filters. A convolutional layer doesn't care about the size of the input image. It just does the convolutions and present a resulting image based on the size of the input image. (Search for some illustrated tutorials about convolutions if this is unclear)
* now the weights in a dense layer are totally dependent on the input size. It's one weight per element of the input. So this demands that your input be always the same size, or else you won't have proper learned weights.

Because of this, removing the final dense layers allows you to define the input size (see in documentation). (And the output size will increase/decrease accordingly).

But you lose the interpretation/classification layers. (You can add your own, depending on your task)

You saw Transfer Learning, and how you can take an existing model, freeze many of its layers to prevent them being retrained, and effectively 'remember' the convolutions it was trained on to fit images.

You then added your own DNN underneath this so that you could retrain on your images using the convolutions from the other model.

You learned about regularization using dropouts to make your network more efficient in preventing over-specialization and this overfitting.

Using dropouts!

Another useful tool to explore at this point is the Dropout.

The idea behind Dropouts is that they remove a random number of neurons in your neural network. This works very well for two reasons: The first is that neighboring neurons often end up with similar weights, which can lead to overfitting, so dropping some out at random can remove this. The second is that often a neuron can over-weigh the input from a neuron in the previous layer, and can over specialize as a result. Thus, dropping out can break the neural network out of this potential bad habit!

WEEK – 4 (Multiclass Classification)

Rock, Paper ,Scissors Multiclass Classification

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

You're coming to the end of Course 2, and you've come a long way! From first principles in understanding how ML works, to using a DNN to do basic computer vision, and then beyond into Convolutions.

With Convolutions, you then saw how to extract features from an image, and you saw the tools in TensorFlow and Keras to build with Convolutions and Pooling as well as handling complex, multi-sized images.

Through this you saw how overfitting can have an impact on your classifiers, and explored some strategies to avoid it, including Image Augmentation, Dropouts, Transfer Learning and more. To wrap things up, this week you've looked at the considerations in your code that you need for moving towards multi-class classification!