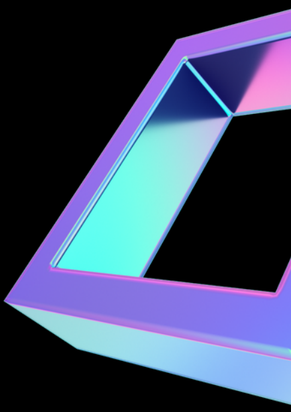
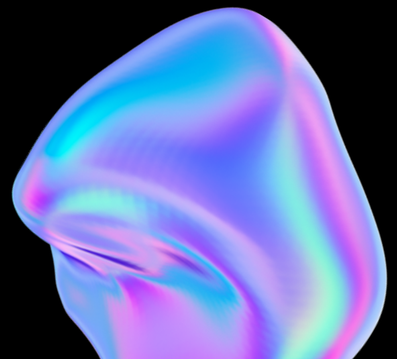
Convolutional Neural Network

23 DECEMBER 2021

>>>TENSORFLOW AUGMENTATION >>>

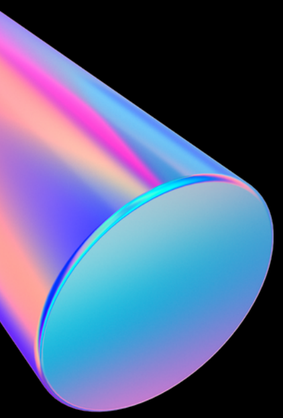
You might only have upright pictures of cats, but if the cat's lying down, or it's on its side, then one ofthe things you can do is rotate the image. So It's like part of the image augmentation, is rotation, skewing, flipping, moving it around the frame, those kind of things. One of the things I find really neat about it, is particularly if you're using a large public dataset, is then you flow all the images off directly, and the augmentation happens as it's flowing. So you're not editing the images themselves directly. You're not changing the dataset. It all just happens in memory. This is all done as part of TensorFlow's Image Generation.

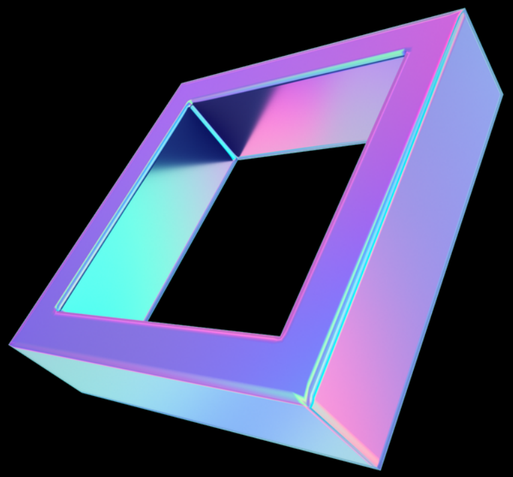


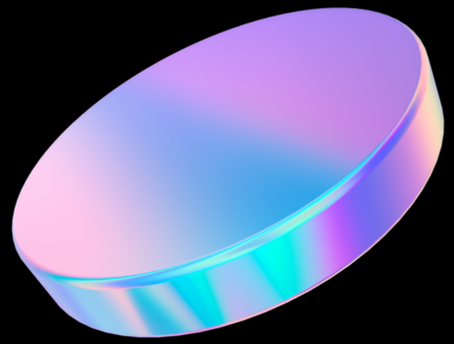


TRNSVER LEARNING

Transfer learning (TL) is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. For example, knowledge gained while learning to recognize cars could apply when trying to recognize trucks.







Apply Convolutional NN to large dataset

One of the nice things with TensorFlow and Keras is that if you put your images into named subdirectories, an image generated will auto label them for you. So the cats and dogs data set you could actually do that and you've already got a massive head start in building the classifier. Then you can subdivide that into a training set and a validation set.

the model.layers API, which allows you to find the outputs and iterate through them, creating a visualization model for each one.

Use ImageDataGenerator to appointed at those folders

if data not normalise use 'rescale =1./255' to normalise

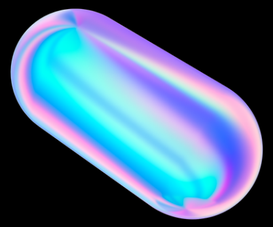
Then call the flow from directory to get a generator object

point that directory for needed datat tehn specify the target size

Batch sizes to be 20. There's 2,000 images, so we'll use a 100 batches of 20 each.

Because we have 2 calsses we use "binary" as a class_methode

In the TensorFlow image augmentation, where the idea is that you're not going to edit the images directly on the drive. As they get float off the directory, then the augmentation will take place in memory as they're being loaded into the neural network for training. So if you're dealing with a dataset and you want to experiment with different augmentations, you're not overriding the data. So to generate a library lets you load it into memory and just in memory, process the images and then stream that to the training set to the neural network we'll ultimately learn on. Training gets little slower, because images prosesing takes cycles.



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.

TRNSVER LEARNING

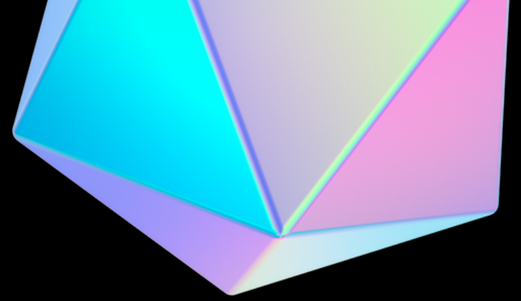
A pre-trained model is a saved network that was previously trained on a large dataset, typically on a large-scale image-classification task. You either use the pretrained model as is or use transfer learning to customize this model to a given task. (It takes an existing model that's trained on far more data, and use the features that that model learned.)

One of the cool things about transfer learning is that it's so simple to implement, in TensorFlow you download a model and then you say set these models as trainable and freeze or lock those other layers and then you just run it.

Dropout

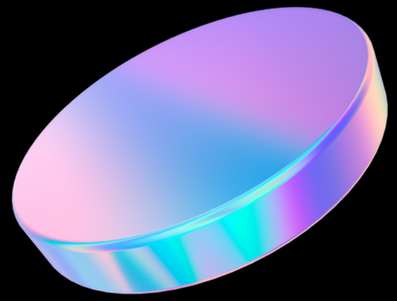
The idea behind Dropouts is that they remove a random number of neurons in your neural network. This works very well for two reasons: The first is that neighboring neurons often end up with similar weights, which can lead to overfitting, so dropping some out at random can remove this. The second is that 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!

Where we use; There's another layer take in Keras called a dropout. And the idea behind the dropout is that layers in a neural network can sometimes end up having similar weights and possible impact each other leading to over-fitting. With a big complex models, that's a risk.



What would the symptom of a Dropout rate being set too high?

The network would lose specialization to the effect that it would be inefficient or ineffective at learning, driving accuracy down



Why do dropouts help avoid overfitting?

Because neighbor neurons can have similar weights, and thus can skew the final training