**Fundamentals of Deep Learning 2025, Apr 06**

Colab notebook showing how to to load a variant of the fruits dataset from Kaggle

<https://colab.research.google.com/drive/1OdqfYdVDhiSY7yQIJ0Jay94em2GPbZ7D?usp=sharing>

fruit classification:

If you didn't receive the Nvidia code via email, you were not part of the first 40 participants who registered with a valid academic address. You can still audit the course and complete the Kaggle challenge here:

<https://www.kaggle.com/t/b4dbb9add11c4da0962b837929799d52>

We will also offer the course again in the future

==

Hello, yesterday you gave a website to access a large stock of images, if someone has the website, I would appreciate it if you could send it.

1. Did you mean <https://try.fiftyone.ai/datasets> ?

**1**

1. [10:13]

Tutorial to set up kagglehub on Google Colab <https://github.com/andandandand/practical-computer-vision/blob/main/docs/kagglehub_setup_colab.md>

==

[setosa.io/ev/image-kernels/](https://setosa.io/ev/image-kernels/)

There are some special matrices that are often used in ML and CV:  
**Sobel Filter:** Detects edges by calculating the gradient magnitude and direction of pixels in an image.  
**Laplacian Filter:** Enhances edges by highlighting regions of rapid intensity changes.  
Yes - we used to do these by hand!   
  
==

09:49

I've watched all your videos and feel comfortable with the classification aspects and fully connected layers, but I'd like to deepen my understanding of the convolutional process specifically. Could you walk through:

1. How exactly do the convolutional filters detect features in an image?
2. What's the intuition behind filter initialization and how they learn during training?
3. How does the concept of receptive fields evolve through a CNN architecture?

10:23

1. Intuition: convolutional networks when run on images just discard the pixel values that do not correlate with the target and highlight the pixel values that do
2. Filters are set completely random when training from scratch. The filters are learned by minimizing the cross entropy loss value (or any other value that is used as loss function) at the final layers of the network. Backpropagation and stochastic gradient descent do the job of defining the values in the convolution filters. When doing inference with a pretrained model, the convolution filters come from the previous training.
3. What do you mean with "evolve"?

==

Traditional image processing uses convolutions with weights 'set by hand'. If we want to blur an image or shapen an image, we already know which set of weights achieve this effect. This doesn't require the usage or training of a convolutional neural network. A set of known image processing kernels with weights set by hand is available here

<https://setosa.io/ev/image-kernels/>

<https://www.image-net.org/>

**ImageNet** is an image database organized according to the [WordNet](http://wordnet.princeton.edu) hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. The project has been [instrumental](https://qz.com/1034972/the-data-that-changed-the-direction-of-ai-research-and-possibly-the-world/) in advancing computer vision and deep learning research. The data is available for free to researchers for non-commercial use.

==

andandandand/images-for-colab-notebooks

<https://github.com/andandandand/images-for-colab-notebooks>

==

Explanation about BatchNorm and turning it off with model.eval()

<https://www.youtube.com/watch?v=3kGE1vCmxag>

==

*transform* changes the np to torch?  
Yes, *ToImage* turns it to a *torchvision.tv\_tensors.Image*, which is a subclass of *torch.Tensor*

==

1. When does it not make sense to use a pretrained model on Imagenet? For example if I am working on medical images, should I look for a model pretrained on medical images (if it exists)?

**1**

A big specialized dataset of medical images is beneficial. However using pretrained weights is better than using purely random weights from untrained models. We benefit a lot from the variety and size of the imagenet dataset. The first layers learn very general features. There are training techniques that exploit this. Look into discriminative learning rates in FastAI and the following paper to get a better sense of this

<https://arxiv.org/pdf/1311.2901> One can also fine tune a model pretrained on Imagenet to perform well on medical images, see the project here:

<https://github.com/pixel-diagnose>

==

Is resizing image require for both training and Inference/prediction?  
The AdaptiveAvgPooling layer makes x and y resizing unnecessary, however the transformations applied on the pretraining have the resizing applied, you can inspect them with   
`from torchvision.models import VGG19\_Weights  
VGG19\_Weights.IMAGENET1K\_V1.transforms()`

+ I would assume that applying the transformation during inference is necessary, because the model was trained with the transformations applied and if you do not use them during inference you are immediately out of distribution?

Correct, the learned filters haven been learned on images of size 224 x 224, this is a reason why the convention of doing resizing persists even though the model doesn't really "need it" during inference. You can find a discussion here

<https://discuss.pytorch.org/t/how-can-torchvison-models-deal-with-image-whose-size-is-not-224-224/51077>

==

mean and std values to transform images are the same for every imagenet classification models?

Yes, these mean and std values come from statistics of the training set of ImageNet (ImageNet is a dataset, not a network, we use ImageNet to train networks). These are computed on the RGB channels, that's why we have three of them. This is explained in this video

<https://youtu.be/0Lz1uDpgVCw?si=dAr2P1U13eUWHMdM>

==

how do we know what transformation will help for the correct reconnection of pytorch ?   
do you have some ressources of different exemple ?  
now we use a transform to have better contrast, but i think for other model we have to do other tricks ?

We use the transformations that were use to "pretrain" the model on ImageNet, this is to make the RGB values of our images be interpreted as similar to the ones on the ImageNet training data. This allows us to do "transfer learning" from the ImageNet pretrained model to our own dataset. We will have a workshop on transfer learning in the coming weeks, however these aspects are explained in the video here

<https://youtu.be/0Lz1uDpgVCw?si=dAr2P1U13eUWHMdM>

==

==

There are no recording for these sessions. The content has already been recorded and edited and posted on Youtube

<https://youtu.be/0Lz1uDpgVCw?si=dAr2P1U13eUWHMdM>

==

Slides for the Image Dataset Curation workshop

<https://docs.google.com/presentation/d/1EHXOoHANv50bEHJ3wKN0nwjlPURbiKLn4ooMJae1LZc/edit?usp=sharing>

[hometogo.com](http://hometogo.com)

==

amazon mechanical turk

https:// [mturk.com](http://mturk.com)

==

MNIST on fiftyone

<https://colab.research.google.com/drive/11dmd1YImqEA5M7_OlXKf16F8QTU8EeA_?usp=sharing>

tutorial:

<https://docs.voxel51.com/tutorials/image_embeddings.html>

==

yes, we can do this with anything that we can feed into a neural network. Here is an example with the BERT embedding model for text

<https://colab.research.google.com/drive/1r0_D346-4ALaVdttV0EAeydgS6_RdtGn?usp=sharing>

Yes, we can do this. Here you can leverage the use of tags

<https://docs.voxel51.com/user_guide/basics.html#tags>

==

1. Paper discussing why embeddings from pretrained networks work so nicely for different tasks (mainly classification and retrieval based on similarity are discussed here)
2. <https://www.csc.kth.se/~azizpour/papers/ha_cvpr14w.pdf>

==

how to run FiftyOne on own computer, not out of colab

<https://docs.voxel51.com/getting_started/install.html>