Lesson 3

Image Transformation/ data augmentation

- default is to grab the center of the image
- everything has to be the same size for the model, commonly this is done as a square
- $.new$ allows for tranformation of existing $DataBlock$ by creating a new one
- $.pad$ pads the image to prevent image distortion due to squishing and stretching ($.squish$)
- $.squish$ is the most efficient, but the distortion can affect the computer's capability to distinguish thin and thick
- $randomresizedcrop$ captures a different part of the image every time randomly with different transformations
  - prevents overfitting
  - $minscale$ denotes minimum %age of pixels in the random crop
- item_tfms happen one image at a time, batch_tfms happen a batch a time on the GPU
- aug_transforms is a standard set of transforms used for images
- fast.ai avoids doing data augmentation on the validation dataset, except for randomresizedcrop where validation set tries to capture the largest center image

---

- unlink() is used to delete the files
- vocab denotes mapping
  - here, shows the label mapping

Deployment

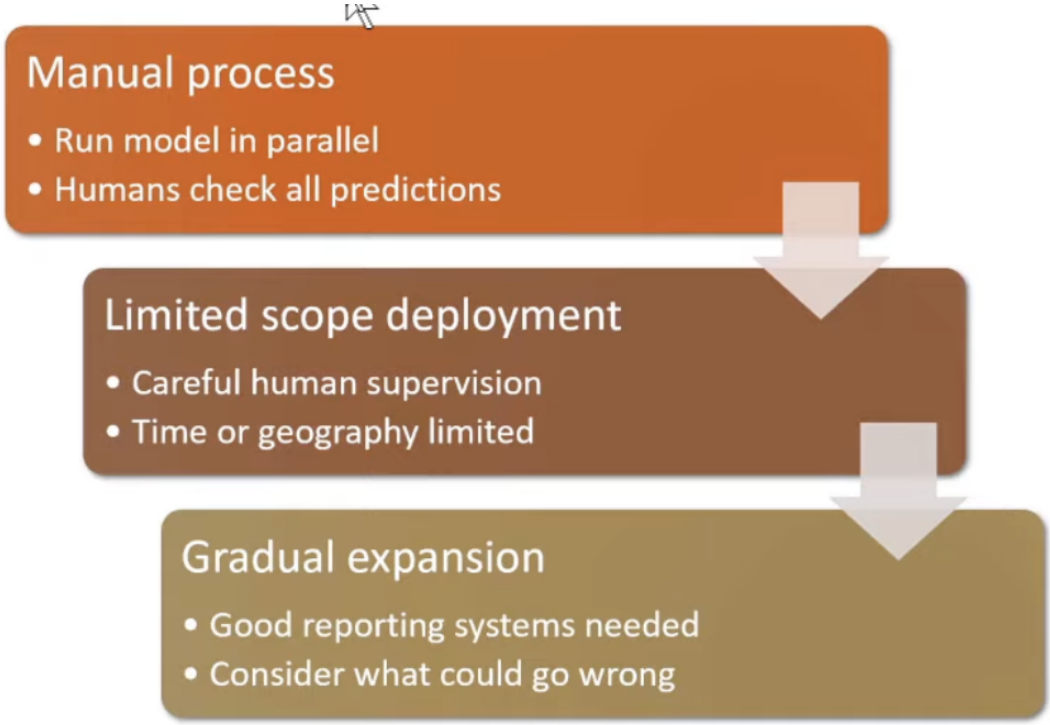
- voila allows the use of markdown and ipywidgets to be presented as a web app

- binder allows the voila notebook URL to be deployed on a free CPU inference server
    - CPU usually works because the inference is being done an image at a time
- For batch of hundreds of images/videos, GPUs are required
- For deployment to mobile phone, have the mobile phone talk to a server where the model is deployed

---

How to avoid disaster

- Be aware of what your data has bias for
- Actionable Auditing — Deb Raji
- Try to gather data to reflects the real world
- Building Machine Learning Powered Applications — Emmanuel Ameisen
- Out of Domain data
    - Data is different from the data it was trained for
    - eg. using stick bear images for a camp detection system that has low res systems with night vision etc
- Domain Shift
    - original training data is no longer relevant over time due to shift in application
- Deployment Strategy



Chapter 4

Starting an MNIST from scratch

- To get details of a function
  - ? to get where it comes from
  - ?? to get the source code
  - doc() to show in docs
- Image is a matrix(array) of numbers with the RGB information
- tensor is a replacement for numpy array

Goal: create a model that can recognize three or a seven

Idea 1: Take average of all 3s in the dataset on a pixel level

- a 28×28 ideal 3 and do the same for 7
- a given picture: is it closer to the three or the seven

- this becomes the baseline model
    - baseline should be easy to program, and used as baseline to compare more sophisticated methods

stacked takes the list of images and stacks them on top of each other to create a tensor of nx28×28 where n is the number of images to make mathematical operations easier

- /255 to make it between 0 and 1 — standard for images
- rank 3 tensor created — 3 axis or 3 dimensions

.mean(0) takes mean over each axis (instead of taking cumulative mean over all axis and giving a singular value)

Difference

- Difference has to be positive, otherwise the positive and the nagative differences can cancel each other out.

Two methods of calculating means

- mean of absolute value of difference — L1 norm — F.l1_loss()
- root mean squared error — L2 norm — F.mse_loss().sqrt()

Computing metrics using broadcasting

- mean((-1,-2)) calculates the mean on the last two axis axis so can be called on multiple matrices at once
- broadcasting: python creates copies of the tensor to do the subtraction as required if the shapes don't match
- loops slow down things

Gradient Descent

- Assume weights for each pixel
- If there is a dot in those places, give it a high weight

Training process — this is gradient descent (not stochastic)

1. Initialize the weight
2. for each image, predict wether its a three or a seven

3. evaluate the performance of the model (loss)
4. calculate gradient → how do the weights change
5. change the weights (step)
6. go back to step 2 and repeat
7. until the process is good enough and you stop

- slope shows the direction of decreasing loss
- _ at the end of a method is an inplace function; it modifies the value in place

---

End to End Gradient Descent Example