How to create a computer model - Fast.AI Video 1.1

START HERE: Within Jupyter Notebook make sure you have the latest version of Fast.AI - !pip install -Uqq fastai

	Image/Object Recognition Model	Segmentation Model	Tabular Analysis Model	Collaborative Filtering Model
Description		Take photos and color in every pixel according to what it is. Classify every pixel.		Basis of most recommendation systems. Which users likes or uses which products, use this to guess what other products those users might like based on find similar users (not demographically, but people who liked the same kinds of products)
Select data	Choose ‘forest’ and ‘bird’ images			
Search for data	Search for ‘forest photo’ and ‘bird photo’			
Download and resize data	Download and resize (don’t need large images) = we have 200 photos each!			
Create a data block Data blocks give Fast.AI all the information it needs to create a computer vision model. It’s also easy to check data in computer vision models Developer Note: You need to think about how to get this data into your model Requirements: What kind of input do you have? What kind of output do you have? What are the items in this model that I’ll be training from? Validation set- Put aside some data to test the accuracy of your model. Fast.AI won’t let you train it without one How do you know the correct label of a photo? Data loaders- things PyTorch iterates through to grab your data at a time. Will feed the training algorithm with a batch of your images at once ➔ NOTE: There’s a data block tutorial under “Data” on docs.fast.ai	Create a data block ● What kind of input do you have?- image ● What kind of output do you have?- category ● What are the items in this model that I’ll be training from?- function. Variable that stores the data (image files in this case) ● Validation set ● How do you know the correct label of a photo? ○ Returns the parent folder of a path ● Data loader: .show_batch- show me a batch of examples of data you’ll be passing into the model, this will show the input (data) and the label	● Data loader classes: There’s very little code required. This doesn’t use data blocks, but you can use data loader classes for the kind of data that’s used a lot and it uses even less code ○ Pass in a function for labeling ○ label_func- grabs a path ○ codes- labels for the code	Data block: utar_data- grabs data and decompresses it for you ● Tabular data loaders- ○ Tell it which columns are categorical ○ Tell it which ones are continuous- take any real number	● Create collaborative filtering data loaders- by downloading and decompressing data ○ Create CollabDataLoaders from a CSV, and pass in the CSV
Data block result	Downloaded 200 photos of forest and 200 photos of birds		● .show_bath- to see the data ○ Type Dispatch- automatically do the right thing for your data no matter what it is. If you call “show_batch” you should get something useful no matter what information you fed it	
Run it (takes about 30 seconds)	Runs through every photo and for the ones that are forest, it’ll learn about what forests look like and vis versa for birds			

Test	Pass in a photo to test it			

Pass data loader subject and the model	• resnet18 is one of fast.ai's built-in models ← they integrated timm library (largest collection of computer vision models in the world). Fast.AI is the first and only framework to integrate this
	  ○ You can use other models at: r.wightman.github.io/pytorch-image-models/results/ (just use the strings)
	• By default one Fast.AI it'll download the weights for you, so you start with a network that can do stuff
		Create a unit learner?	Build a model for this data loader- (title)_learner, in this case tabular_learner	• Create a learner- collab_learner and pass your data
				  ○ Since it's not predicting a category, it's predicting a real numbers, we tell it what's the possible range. The actual range is 1-5, but it's a good idea to go lower than the possible minimum

Fine-tune method	Takes pre-trained weights and adjust them to teach the model the difference between your dataset and what it was originally trained for	Fine-tune	• This time we don't say "fine tune" we say "fit_one_cycle" ○ There are no tabular models because there are generally no pre-trained model that already does what you want it to do because every table of data is different, whereas images have the same general idea	We don't need to fine-tune, because there's no such thing as a pre-trained collaborative filtering model. We could say "one_fit_cycle", but fine-tune is good as well

Use the computer vision model!- This is how you would deploy your model	1. Call .predict and pass an image
	2. It will return:
	  1. "Is it a bird or not?" as a string
	  2. "Is it a bird or not?" as an integer
	  3. Probability that it's a bird or not a bird			

Other Notes:

1. The Deep Learning community has found a small number of models that will work for nearly any application. Fast.ai will create the right type of model most of the time. Tweaking NNL architecture almost never comes up.
2. Most computer vision architectures needs all the inputs that you trained to be the same size
3. For any fast.ai model you can call .show_results