Lesson 1

- Google's Pathways Language Model (PaLM) - Google AI blog.
- Universal language model fine-tuning for text classification

1. You'll notice that now it's very easy to you know classify an object as a bird or not. A decade back it was very difficult. Reason? In a project at Stanford to detect breast cancer cells, they got a cross-disciplinary team of Mathematicians, Computer Scientists, doctors to write a lot of code, with a lot of Maths to firstly build features. Then feed it into a logistic regression classifier.
2. What does a neural network does? We don't have to build features by ourselves, it take cares of it and gives us result in very few lines of code.
3. In this course we will spend time building image-based algos, and it's important to know that image based algos aren't just for images and in fact this is going to be a general theme. There are cases of students using image recognizer to classify sounds, taking time-series and turning it into images and then use image classifiers.
4. We'll be using PyTorch since it's better than TensorFLow from the looks of fundaments(not my words, instructor's). But it requires a lot of hairy code to implement simple things, so we'll be using the FastAI libraray nuilt on top of pytorch. PyTorch is designed to be a strong foundation to build things on top of, like fast.ai . But as we go deeper and deeper into the course, we'll see a lot more of Pytorch.
5. Let's talk about data blocks a bit. From project to project, input data is one thing that changes for sure. So you'll need to know?
    - what kind of input do we have?
    - what kind of output is there?
    - what are the items in the model?
    - what am I actually going to be looking at to look train from? this is a function:- get_images_files, a function which returns a list of all of the image files in a path based on extension.
    - Random splitter splits your dataset for testing/validation. Fast AI won't allow you train the model without that.
    - get_y is a function that simply returns the parent folder of a path to our labels
    - Most computer vision architecture need all of your inputs to be of the same size, "item transforms" does that for you.
    - From here we create an important class called "dataloaders" , pytorch uses it to iterate through to grab a bunch of data. The way it can do it so fast is by using GPU. Dataloaders feeds your training algo with a bunch of data at once, this bunch is called

"batch" or "mini-batch" .

```
dls = DataBlock(
    blocks=(ImageBlock, CategoryBlock),
    get_items=get_image_files,
    splitter=RandomSplitter(valid_pct=0.2, seed=42),
    get_y=parent_label,
    item_tfms=[Resize(192, method='squish')]
).dataloaders(path)

dls.show_batch(max_n=6)
```

6. Use docs.fast.ai for tutorials on different stuff.
7. A critical concept in Fast AI is called a "learner". It's something which combines a model and the data we will use to train it with, and that's why you have to pass in two things.

```
learn = vision_learner(dls, resnet18, metrics=error_rate)
learn.fine_tune(3)
```

8. Fast AI has an integrated library by Ross Wightman called "timm" (the pytorch image models), which is the largest collection of computer vision models in the world. You can see the documentation at timm . Having said that , the model family "resnet18" is gonna be fine for almost all problems. And you'll see that this works very fast when implemented. The Reason is that someone else has already trained this model with 1 million images of over 1000 different types, something called the "imageNET" dataset., and they made those weights available. So when using this at FastAI, it'll automatically download it first , so when you start using the neural network that can do a lot of stuff.
   Also, fastai has this unique fine-tuned method. It takes those pre-trained weights and it adjusts them in a really carefully controlled way to just teach the model the differences between your dataset and what it was originally trained for. That's called "fine-tuning".
9. To deploy the model or prediction you can use "predict" and the pass the image.
10. That was for CV model, what about other types of model? Well within CV, there's more than just image recognition. e.g. there's segmentation : we take in an image and color each pixel according to the object, like imagine cars on the roads. Road in one color, each car in a different color, and so on..

```
path = untar_data(URLs.CAMVID_TINY)
dls = SegmentationDataLoaders.from_label_func(
    path, bs=8, fnames = get_image_files(path/"images"),
    label_func = lambda o: path/'labels'/f'{o.stem}_P{o.suffix}',
    codes = np.loadtxt(path/'codes.txt', dtype=str)
)

learn = unet_learner(dls, resnet34)
learn.fine_tune(8)
```

You can see that we aren't even using datablocks here, they are very intermediate,flexible approach that you can take in handling almost any kind of data, but for the kinds of data that occur a lot, you can use these special data loaders classes as shown above.

11. What about stepping away from computer vision? So perhaps the most widely used kind of module used in industry is tabular analysis. So taking things like, spreadsheets and database tables and trying to predict columns of these.

Tabular analysis - income prediction

```
from fastai.tabular.all import *
path = untar_data(URLs.ADULT_SAMPLE)

dls = TabularDataLoaders.from_csv(path/'adult.csv', path=path, y_names="salary
    cat_names = ['workclass', 'education', 'marital-status', 'occupation', 're
    cont_names = ['age', 'fnlwgt', 'education-num'],
    procs = [Categorify, FillMissing, Normalize])
```

"untar_data" downloads some data and decompresses it for you. So we create tabular dataloaders this time, we have to tell which of the columns are categorical and which ones are continuous.

```
learn = tabular_learner(dls, metrics=accuracy)
learn.fit_one_cycle(2)
```

Also we said fit not fine_tune to fit the data in one cycle, that's because for tabular data, there won't be any pre-trained model to fine tune.

12. **Collaborative filtering-recommendation system:-** It is the basis of most recommendation systems today, we take dataset that says: which users liked which products/used which products and then we use it to guess what other products those users might like based on finding similar users, and what those similar users like.

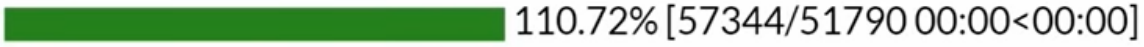
```
from fastai.collab import *
path = untar_data(URLs.ML_SAMPLE)
dls = CollabDataLoaders.from_csv(path/'ratings.csv')
```

```
dls.show_batch()
```

	userId	movieId	rating
0	176	4306	3.5
1	463	4886	4.0
2	313	3578	4.5
3	242	2858	5.0
4	17	608	3.5
5	382	356	3.0
6	547	480	3.0
7	119	457	3.0
8	358	2918	4.0
9	461	2571	4.5

Then we create a learner and use fine_tune(), we wanted to fit the data in many cycles:

```
learn = collab_learner(dls, y_range=(0.5,5.5))
learn.fine_tune(10)
```



13. Here's a list of things that deep learning or methods heavily using deep learning are the best in the world:
    - **Natural language processing (NLP)**
      Answering questions; speech recognition; summarizing documents; classifying documents; finding names, dates, etc. in documents; searching for articles mentioning a concept
    - **Computer vision**
      Satellite and drone imagery interpretation (e.g., for disaster resilience), face recognition, image captioning, reading traffic signs, locating pedestrians and vehicles in autonomous vehicles
    - **Medicine**
      Finding anomalies in radiology images, including CT, MRI, and X-ray images;

counting features in pathology slides; measuring features in ultrasounds; diagnosing diabetic retinopathy

- **Biology**
  Folding proteins; classifying proteins; many genomics tasks, such as tumor-normal sequencing and classifying clinically actionable genetic mutations; cell classification; analyzing protein/protein interactions
- **Image generation**
  Colorizing images, increasing image resolution, removing noise from images, converting images to art in the style of famous artists
- **Recommendation systems**
  Web search, product recommendations, home page layout
- **Playing games**
  Chess, Go, most Atari video games, and many real-time strategy games
- **Robotics**
  Handling objects that are challenging to locate (e.g., transparent, shiny, lacking texture) or hard to pick up
- **Other applications**
  Financial and logistical forecasting, text to speech, and much, much more..