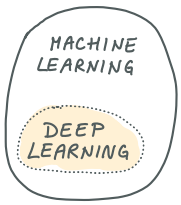
CH 1 DEEP LEARNING JOURNEY

Deep learning for Coders with fastai & PyTorch (source)

Deep learning → a computer technique to extract and transform data by using multiple layers of neural networks

↳ lots of — data ✕
math ✕
expensive computers ✕



NLP · ROBOTICS · MEDICINE · BIOLOGY · IMAGE GENERATION · CV · RECOMMENDER SYSTEMS · FINANCE · PLAYING GAMES ...

HISTORY OF DEEP LEARNING

1943 Warren McCulloch, Walter Pitts ⟫ mathematical model of artificial neuron: simple addition + thresholding

‖ "A logical calculus of the ideas immanent in nervous activity."

↳ Frank Rosenblatt → ability to learn
1st device ⇒ Mark I Perceptron
{recognizes simple shapes}

‖ "The design of an inteligent automaton."

↳ Minsky + Papert ⇒ ‖ Perceptrons
✕ single layer can't learn XOR
✓ use multiple layers ✓

XOR : inequality function

A	B	A XOR B
1	0	1
0	1	1
1	1	0
0	0	0

↳ ~ 1980 ⇒ most models with 2nd layer of neurons

↳ 1986 Rumelhart, et. all ‖ Parrallel Distributed Processing (PDP)

- traditional computer programs work differently from brains
- PDP approach closer than other frameworks

- a set of processing units
- state of activation
- output function for each unit
- pattern of connectivity among units
- propagation rule ⇒ network
- activation rule
- learning rule
- environment

modern neural network can handle these requirements!

- adding additional layer of neurons ⇒ can approximate any mathematical function
BUT too big + too slow ⇒ not useful.
↳ in the last decade applied + widely used

extra layers
+
computer hardware
+
data availability

HOW TO LEARN

teach the whole game!

‖ "Making learning whole"
‖ "A mathematician's lament"

- start with smaller projects
- be playfull & curious

PERSEVERE

APPLY WHAT YOU LEARN TO A PERSONAL PROJECT

experiment on a subset of data ▶ run on full data only when understanding is complete

P1

MACHINE LEARNING

traditional programing → machine learning

1949 Arthur Samuel (IBM) ⇒ machine learning

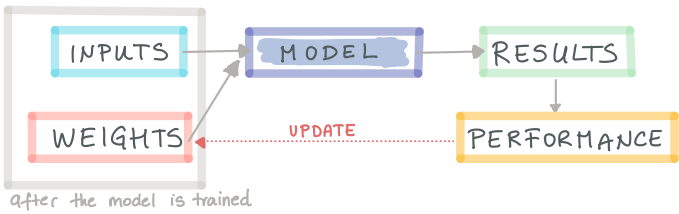
replace exact steps with examples of the problem

weight assignment ⇒ **weights** (⇐ variables)

- particular choice of values for those variables define how the program will operate

> "Artificial Intelligence : A frontier of automation"

- learning is automatic when the adjustment of weights is automatic



- need a flexible function to solve any problem by varying its weights only ⇒ NEURAL NETWORK

UNIVERSAL APPROXIMATION THEOREM : NN can solve any problem to any level of accuracy ; in theory.

2. how to update the weights automatically
   ↳ stochastic gradient descent (SGD) CH4

DEEP LEARNING JARGON

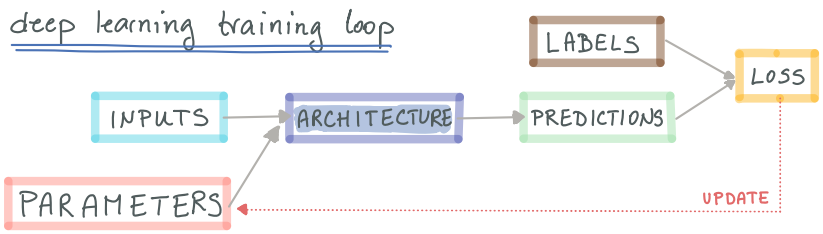
ARCHITECTURE ▸ functional form of the model (template of the mathematical function)

PARAMETERS ▸ weights

PREDICTIONS ▸ results of the model computed from the independent variable (without the labels)

LOSS ▸ measure of performance depends on the predictions and correctness of the labels ↔ targets (dependent variable)

deep learning training loop



★ KEY TO DEEP LEARNING

determining how to fit the parameters of a model to get it to solve the underlying problem

LIMITATIONS

- model cannot be created without data
- model can learn only on the patterns seen in the input data (i.e. training data)
- creates only predictions not recommended actions
  ↳ can create gaps between goals and model capabilities
- to train we need input data + labels
- problems with lack of labeled data
- model vs environment ⇒ feedback loops
  ↳ biased model ⇒ biased results ⇒ using the model enhaces the bias further. (e.g. predictive policing)

FAST.AI IMAGE RECOGNIZER

```
# get the latest version of fastai
!pip install -Ugq fastai
```

```
from fastai.vision.all import *
```
provides all the classes and functions for computer vision (CV) models

```
path = untar_data(URLs.PETS)/'images'
```
download and extract (if it doesn't exist locally) a fastai dataset; returns a path object {pathlib library}

```
def is_cat(x): return x[0].isupper()
```
function that defines how to extract the label from the filename

```
dls = ImageDataLoaders.from_name_func(
    path,
    get_image_files(path),
    valid_pct=0.2,
    seed=42,
    label_func=is_cat,
    item_tfms=Resize(224))
```
DEFINE THE DL DATASET (data type)

- valid_pct=0.2 (default value): allocate 20% of input data for validation set (used to measure accuracy of the model)
- seed=42: sets the random seed so that we get the same validation set on each run (retrain)
- label_func=is_cat: how to define labels (CV problems: label is part of a filename or path/often parent folder name)
- item_tfms=Resize(224): each item is resized to 224×224 pixels.
  - why 224? standard size
  - batch_tfms: applied to a batch of items at a time using the GPU.
  - ↑ size ⇒ better model results ⇒ speed + memory consumption ↑

★ fastai always shows model's accuracy using only validation set

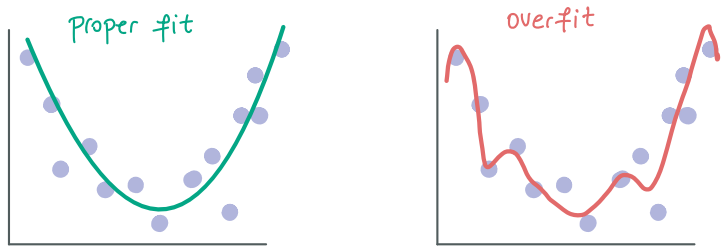
more training iterations
↳ model memorizes certain parts of data
- training accuracy ↑
- validation accuracy ↑ (initial time period)
- validation accuracy ↓ (after the initial time period)

▶ **OVERFITTING**

model fails to generalize on unseen data

validation accuracy ↓ as training progresses

▶ most important and challenging issue in ML



▶ not enough data + long training time
⇒ accuracy ↓

CLASSIFICATION vs. **REGRESSION**

predicts **class** or **category** | predicts one or more **numeric quantities**

Example: studying for the exam
- select n problems to study (input data)
- save n·20% problems to validate your knowledge
- studying time (training) ↑ ⇒ training accuracy ↑, validation accuracy ↑
- slowly one starts to memorize problems & technique ⇒ validation accuracy ↓ OVERFITTING ⇒ final test scores not great!

```
learn = vision_learner(
    dls,
    resnet34,
    metrics = error_rate)

learn.fine_tune(1)
```
- vision_learner: create a convolutional neural network
- dls: DL dataset created above
- resnet34 (PRETRAINED MODEL): CNN architecture; 34 denotes the number of layers (other options: 18, 50, 101, 152)
  - more layers ⇒ training time ↑ ⇒ potential overfitting
  - use more data
- metrics = error_rate: tells us the % of images classified incorrectly in the validation set
- fine_tune(1): defines how to fit a model; 1 = number of epochs
  - fine_tune instead of fit since we are using a pretrained model

layers ↑ ⇒ epochs ↓
! depends on data quantity

METRIC: a function that measures the quality of the model's predictions using the **validation set**.

▶ choose metric that is applicable to the problem and easy to communicate and understand

LOSS: measure of performance during **training** ⇒ used to update the weights automatically.

▶ choose loss that is easy for SGD

ACCURACY: 1 - ERROR RATE

EPOCH: one complete pass through the dataset

PRETRAINED MODEL: a model that has weights that have been trained on another dataset.

- using a pretrained model for a task different than what was originally trained for is called TRANSFER LEARNING

- nearly always use a pretrained model
- vision_learner()
  - removes the last layer
  - replaces it with one or more new layers with randomized weights with appropriate size → head

since it is specifically customized to the original training task

---

- ☐ many non-image tasks can be represented as image ▶ use CNN architecture to build models.
  - sound, time series, computer mouse behavior, malware classification
- ☐ see book pg 36-39 for more examples

VALIDATION & TEST SETS

- always split the dataset into training & validation sets
- if we train more than once on these sets (always the case)
  - ⇒ slowly the model/we learn in the process

choosing the learning rate
model architecture
data augmentation
....
hyperparameters

- construct the test set (totally hidden dataset) used to evaluate the FINAL model

	TRAINING PROCESS	TESTING
TRAINING DATA	fully exposed	not exposed
VALIDATION DATA	partially exposed	not exposed
TEST DATA	not exposed	fully exposed

} should have enough data to ensure that we get good estimates of accuracy

⚠ validation and test sets have to be representative of the future data
e.g. choosing a random validation & tests for time series data is structually wrong ⇒ not representative
☐ see more examples in the book pg. 51.