

THE CODING-WORKSHOP WILL
START SOON!

PLEASE MUTE YOUR MIC AND
DISABLE YOUR CAMERA!

THE SESSION WILL BE
RECORDED



CODING AI

Presentation

~20 min

Live coding

~50 min

Introduction to PyTorch
By Kevin Trebing

ABOUT ME

- Education
 - 2013-2017: B.Sc. Cognitive Science in Osnabrück, Germany
 - 2018-2020: M.Sc. Artificial Intelligence in Maastricht, Netherlands
- Worked on several AI-projects in different companies:
 - Predictive Maintenance
 - Weather prediction
 - Evolutionary game design
 - Automatic menu-card classification
 - ... more
- Extensive use of PyTorch in Master's degree
- Contact: Kevin.Trebing@gmx.net



WHAT IS PYTORCH?



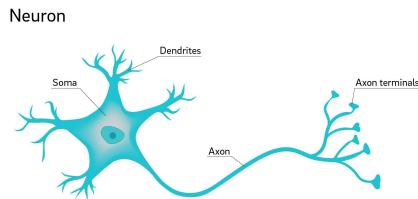
- Python library with C++ backend
- A library to replace NumPy in order to use GPUs
 - Similar functions
 - Similar syntax
- Deep learning library
 - Easy implementation of neural networks
 - Easy training of neural networks
 - Fully customizable (more advanced)
- Good intro resource: [Deep Learning with PyTorch](#)

WHY PYTORCH?

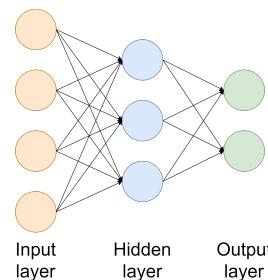
- Very Python-like:
 - Classes
 - If-else conditions
 - Loops
- Big and great community
- A lot of users → easy to get help (Stackoverflow, Forums)
- Gets used by many researchers → newest models available
- Good documentation (looking at you Tensorflow!)

(ARTIFICIAL) NEURAL NETWORK - RECAP

- Not real neurons (just the idea)



- Artificial** neurons are values (weights) that are adapted
- Combining multiple neurons results in a neural network
 - E.g: Multilayer Perceptron (MLP)



HOW TO TRAIN A NEURAL NETWORK? - DATASET

Suppose we have examples:

Image



Label

Dog

Dog

Cat

Cat

Very hard task already!

RELEVANT XKCD

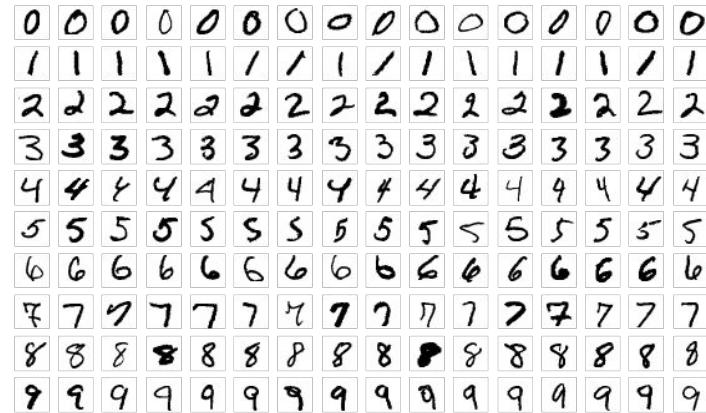


IN CS, IT CAN BE HARD TO EXPLAIN
THE DIFFERENCE BETWEEN THE EASY
AND THE VIRTUALLY IMPOSSIBLE.

HOW TO TRAIN A NEURAL NETWORK? - START EASY

Recognize handwritten digits

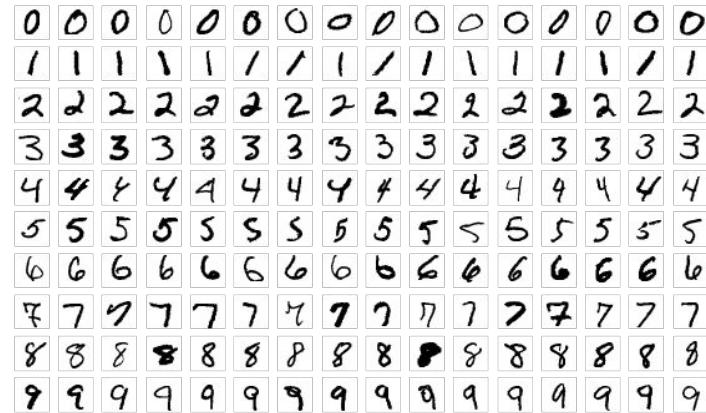
- MNIST-dataset:
 - Input: 28x28 pixel images
 - Output: Label



HOW TO TRAIN A NEURAL NETWORK? - START EASY

Recognize handwritten digits

- MNIST-dataset:
 - Input: 28x28 pixel images
 - Output: Label



HOW TO TRAIN A NEURAL NETWORK? - NORMALIZE DATA

- Data should be normalized → In the range between 0 and 1 or -1 and 1
- Reduces chance of saturated neurons → unsaturated is better for learning

HOW TO TRAIN A NEURAL NETWORK? - NORMALIZE DATA

- Data should be normalized → In the range between 0 and 1 or -1 and 1
- Reduces chance of saturated neurons → unsaturated is better for learning
- Can be done by calculating mean and standard deviation:

$$\frac{X - \mu_x}{\sigma_x}$$

- Or min-max scaling:

$$\frac{X - \min(x)}{\max(x) - \min(x)}$$

HOW TO TRAIN A NEURAL NETWORK? - LOSS FUNCTIONS

How to determine how good/bad a model's prediction is?

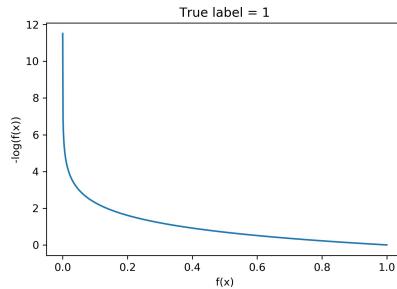
-> Use a function that compares the true label and predicted one

HOW TO TRAIN A NEURAL NETWORK? - LOSS FUNCTIONS

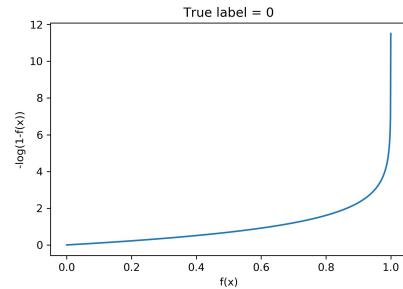
How to determine how good/bad a model's prediction is?

-> Use a function that compares the true label and predicted one

- Binary classification
 - Log-loss



$$-\log(f(x))$$



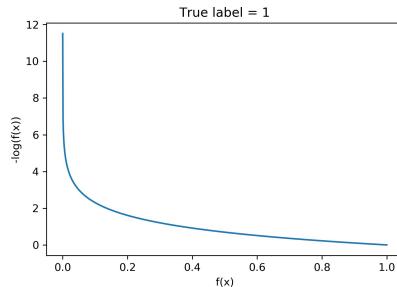
$$-\log(1 - f(x))$$

HOW TO TRAIN A NEURAL NETWORK? - LOSS FUNCTIONS

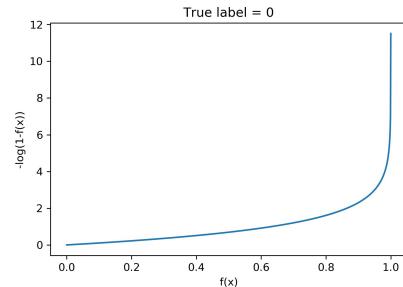
How to determine how good/bad a model's prediction is?

-> Use a function that compares the true label and predicted one

- Binary classification
 - Log-loss
- More classes
 - Cross-entropy-loss
- Regression task
 - L1 loss $|y_{pred} - y_{true}|$
 - L2 loss $|y_{pred} - y_{true}|^2$



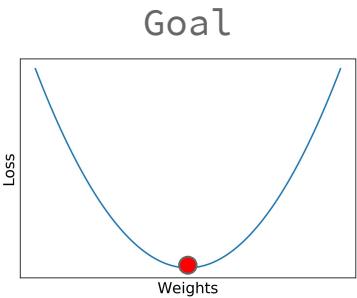
$$-\log(f(x))$$



$$-\log(1 - f(x))$$

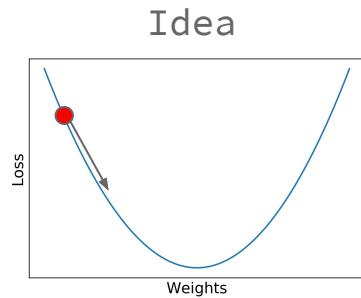
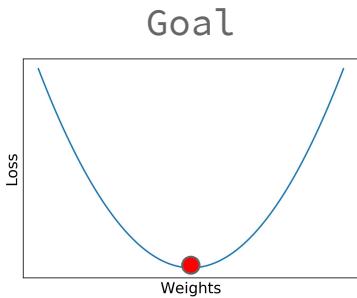
HOW TO TRAIN A NEURAL NETWORK? - OPTIMIZERS

Minimize loss and adapt weights with backpropagation ->
“optimize”



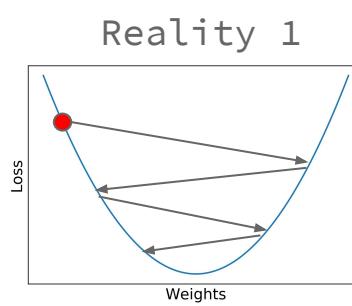
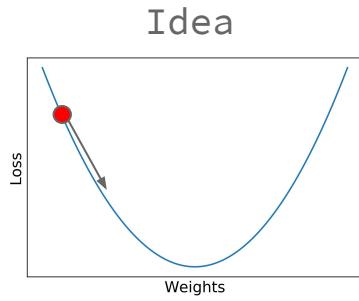
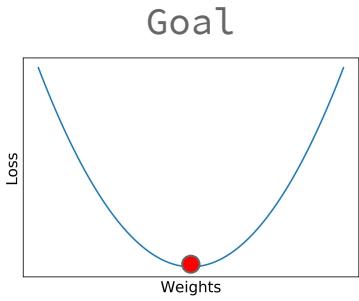
HOW TO TRAIN A NEURAL NETWORK? - OPTIMIZERS

Minimize loss and adapt weights with backpropagation ->
“optimize”



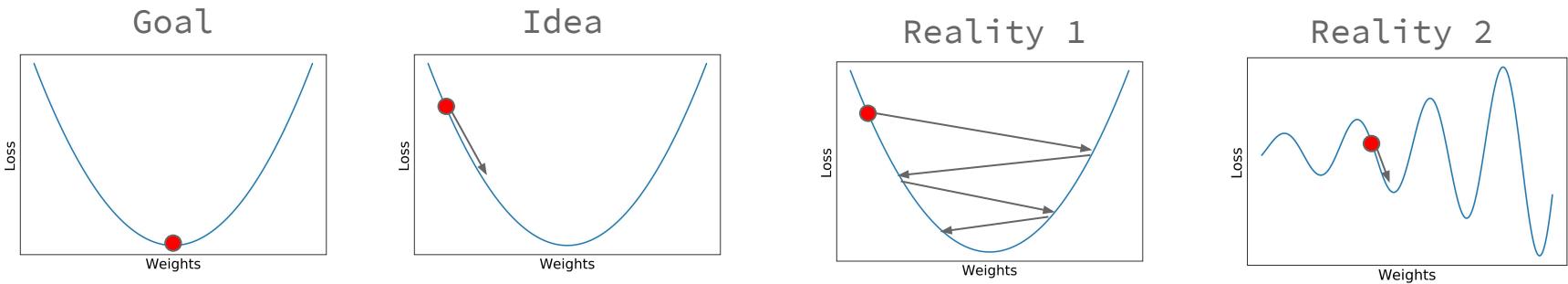
HOW TO TRAIN A NEURAL NETWORK? - OPTIMIZERS

Minimize loss and adapt weights with backpropagation ->
“optimize”



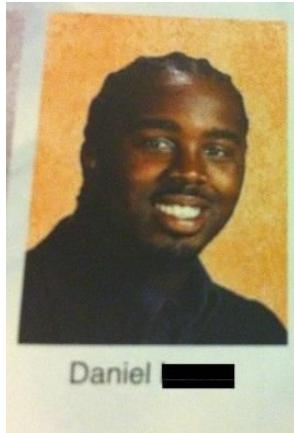
HOW TO TRAIN A NEURAL NETWORK? - OPTIMIZERS

Minimize loss and adapt weights with backpropagation ->
“optimize”

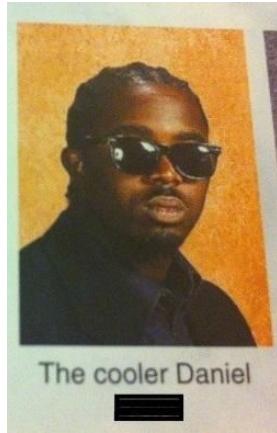


Important parameters: **optimizer** and **learning rate**

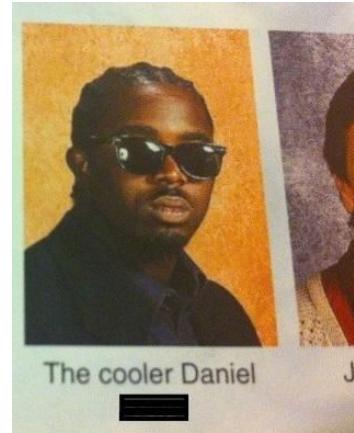
CHOOSE YOUR OPTIMIZER



SGD



RMSprop



Adam

...

Optimizer receives the model's parameter and learning rate

SUMMARY - WHAT DO WE NEED?

1. **Dataset**
 - a. Train
 - b. Validation
 - c. Test
 2. Neural network model
 3. Loss function
 4. Optimizer
 5. Hyperparameters:
 - a. How long to train - epochs
 - b. Batch size
 - c. Learning rate
- from torch.utils.data import Dataset, DataLoader
- from torch import nn
- from torch.nn import CrossEntropyLoss
- Or from torch.nn.functional import cross_entropy
- from torch.optim import Adam, RMSprop, SGD

DATASET - CLOSER LOOK

Train network to generalize -> able to predict/classify unseen samples

Whole dataset



DATASET - CLOSER LOOK

Train network to generalize -> able to predict/classify unseen samples

Whole dataset



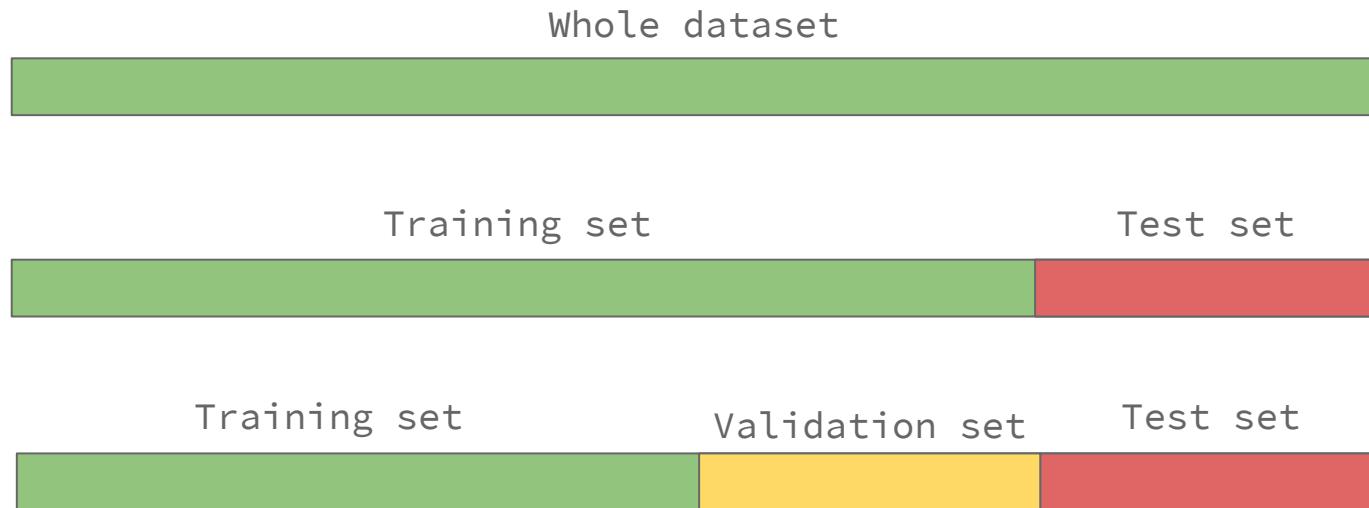
Training set

Test set

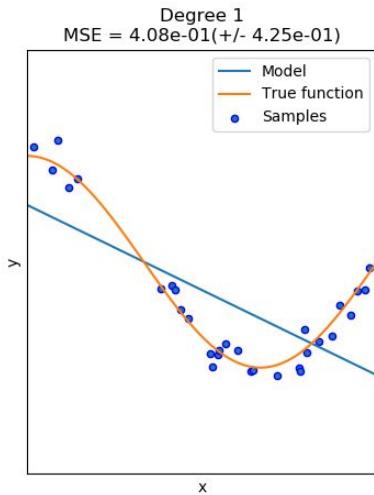


DATASET - CLOSER LOOK

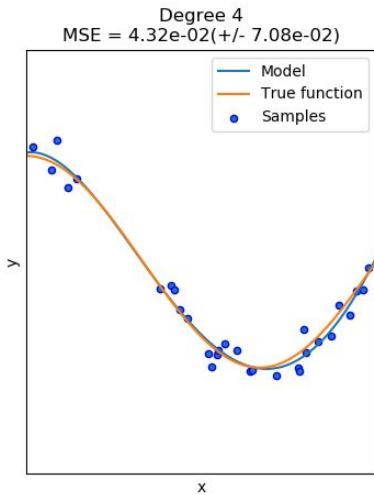
Train network to generalize -> able to predict/classify unseen samples



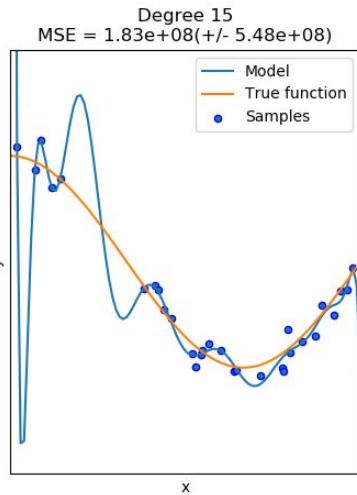
WHY THE HASSLE?



Underfit

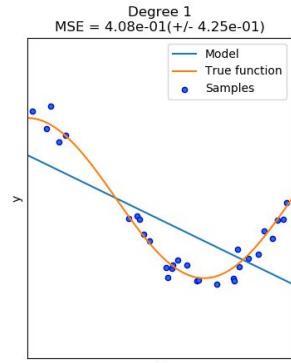


Good fit

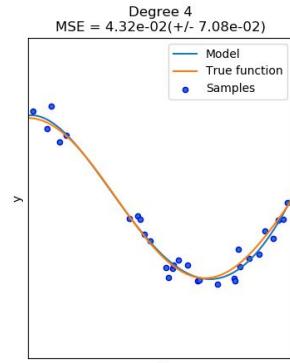


Overfit

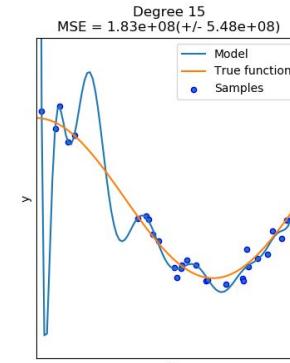
WHY THE HASSLE?



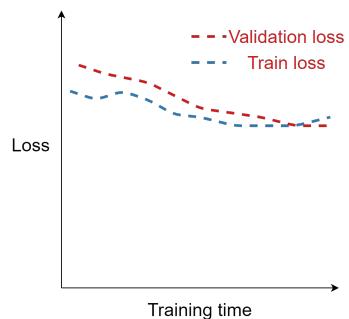
Underfit



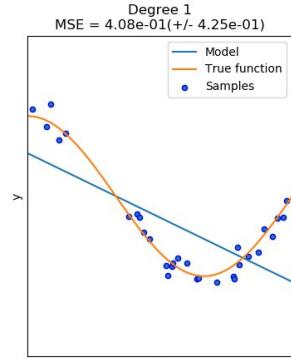
Good fit



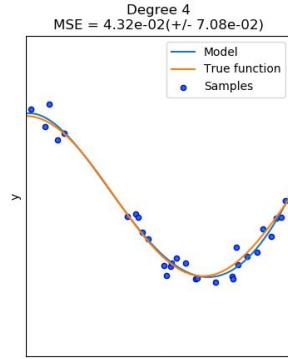
Overfit



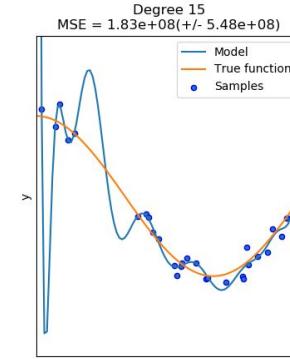
WHY THE HASSLE?



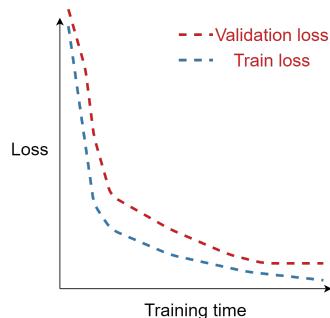
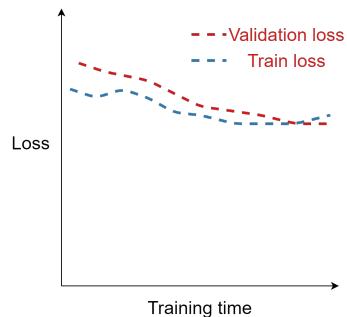
Underfit



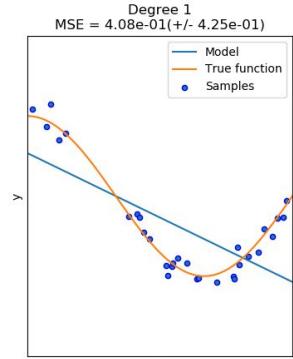
Good fit



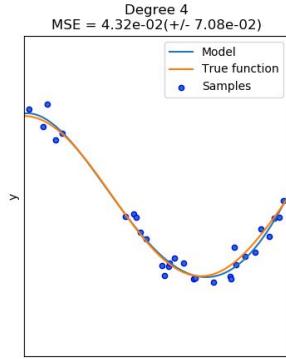
Overfit



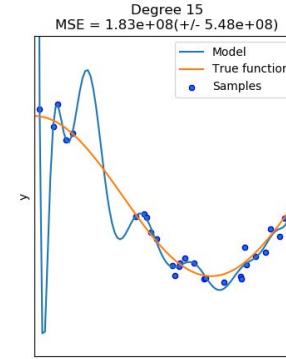
WHY THE HASSLE?



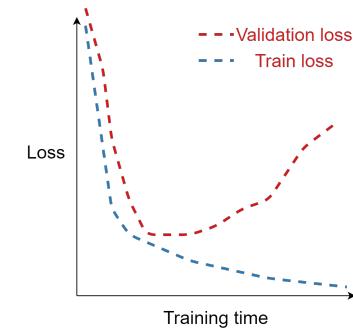
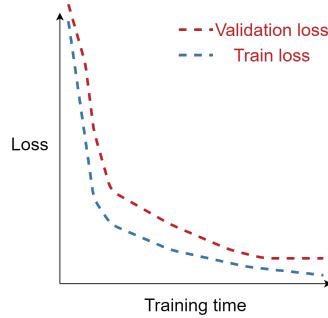
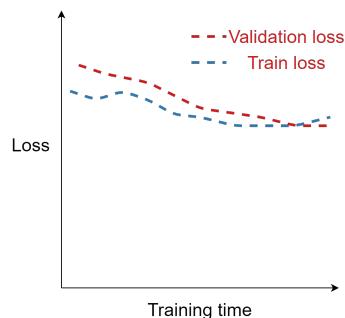
Underfit



Good fit



Overfit



SUMMARY (AGAIN)- WHAT DO WE NEED?

1. Dataset
 - a. Train
 - b. Validation
 - c. Test

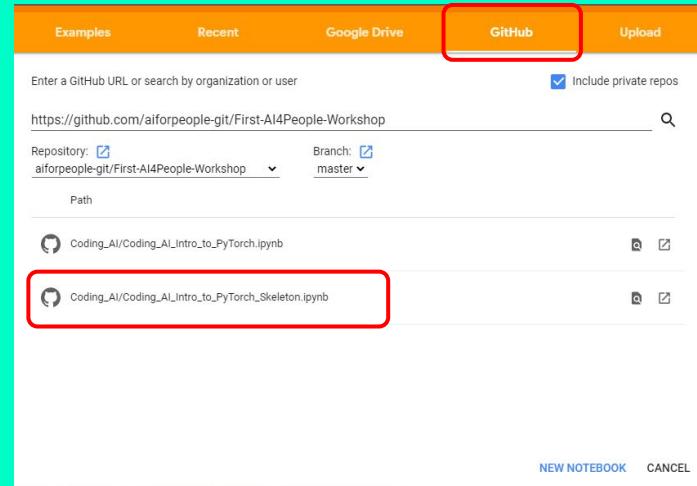
from torch.utils.data import Dataset, Dataloader
from sklearn.model_selection import train_test_split
Or from torch.utils.data import random_split
2. Neural network model
from torch import nn
3. Loss function
from torch.nn import CrossEntropyLoss
Or from torch.nn.functional import cross_entropy
4. Optimizer
from torch.optim import Adam, RMSprop, SGD
5. Hyperparameters:
 - a. How long to train - epochs
 - b. Batch size
 - c. Learning rate

NOW TO THE CODING!

An example of how to write code to train a neural network with PyTorch

1. Go to: <https://colab.research.google.com/>
2. Open the code-skeleton at:

<https://github.com/aiforpeople-git/>



THANK YOU FOR
LISTENING!
QUESTIONS?

ADVANCED PYTORCH THINGS

- Learning rate scheduler
- Tensorboard logging
- Saving the trained model
- Loading the saved model

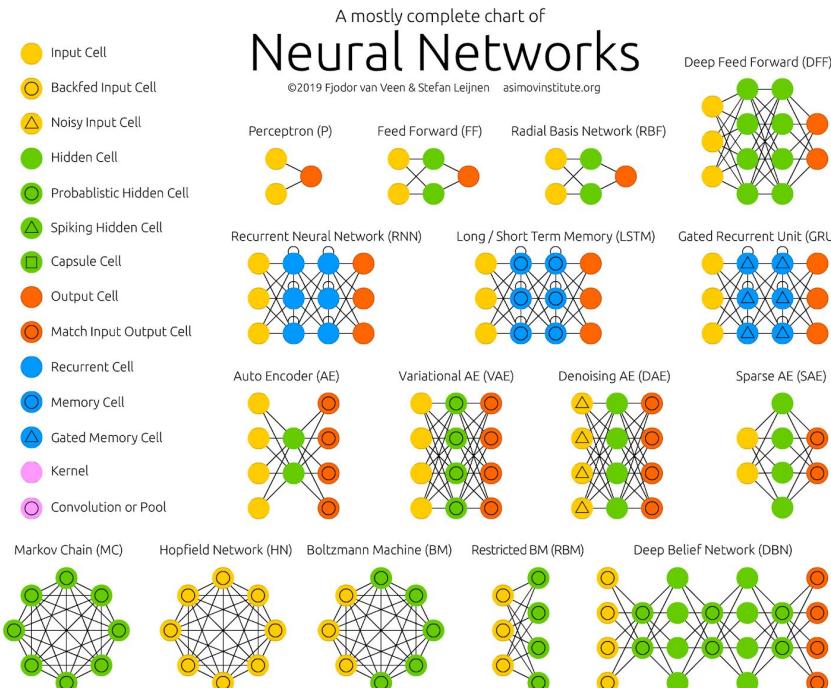
from torch.optim.lr_scheduler

from torch.utils.tensorboard import SummaryWriter

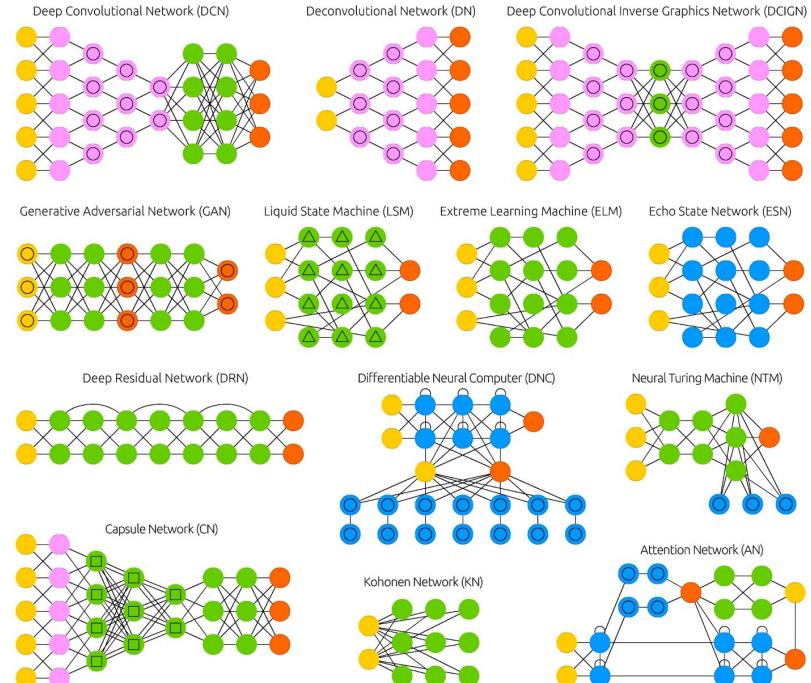
torch.save({'state_dict': model.state_dict}, 'saved_model.pt')

model.state_dict = torch.load('saved_model.pt')['state_dict']

OH, THE POSSIBILITIES!



Taken from: <https://www.asimovinstitute.org/neural-network-zoo/>



REFERENCES

Axon: <https://medicalxpress.com/news/2018-07-neuron-axons-spindly-theyre-optimizing.html>

Cat image 1:

<https://timesofindia.indiatimes.com/life-style/relationships/pets/5-things-that-scare-and-stress-your-cat/articleshow/67586673.cms>

Cat image 2: <https://www.wired.com/story/grumpy-cat-obit/>

XKCD: <https://xkcd.com/1425/>

MNIST: https://en.wikipedia.org/wiki/MNIST_database

Over and underfit: https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html