Machine Learning for Jet Physics

RTG Student Lecture

Peter Sorrenson

Contents

Lecture 1 – ML history and present (13.05)

- Machine learning history and important models

Lecture 2 – Applications to jet physics (20.05)

- Symmetries
- Representation learning
- Anomaly detection

Lecture 3 – Tricks and trends (03.06)

- Practical tips for a successful ML project
- Current trend: large language models

Lecture 1

ML history

- Main idea of ML: learn statistical patterns from data in order to perform tasks without explicit programming
    - Driving force: continually dropping cost of computation and data (*Two Centuries of Productivity Growth in Computing*, Nordhaus 2007)

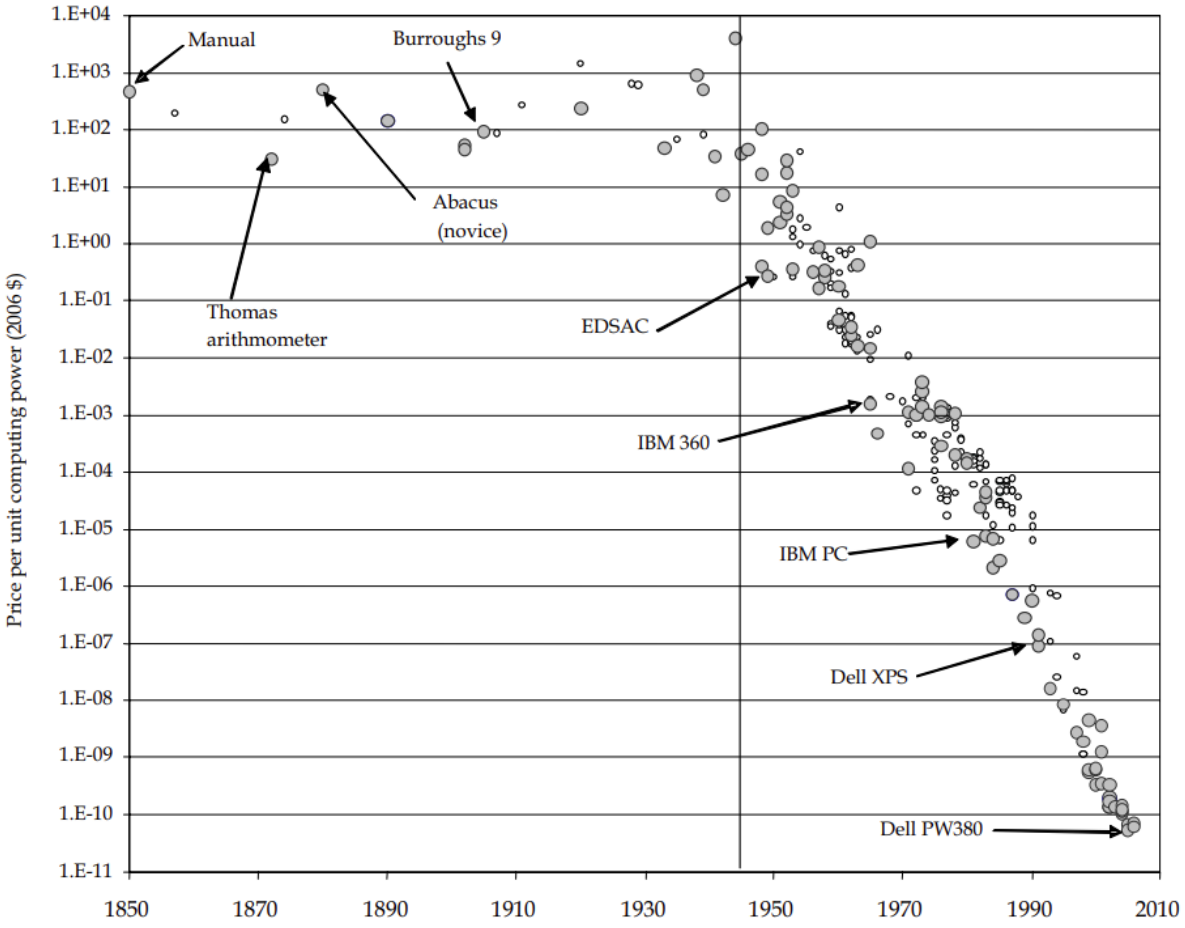
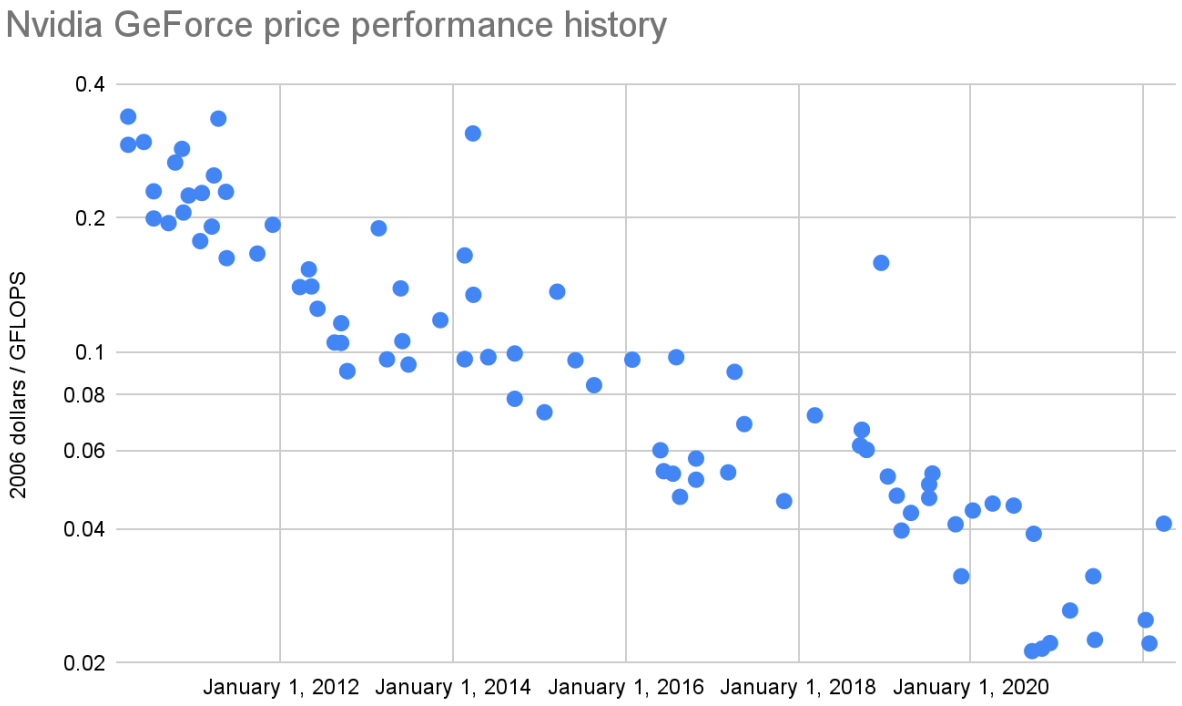
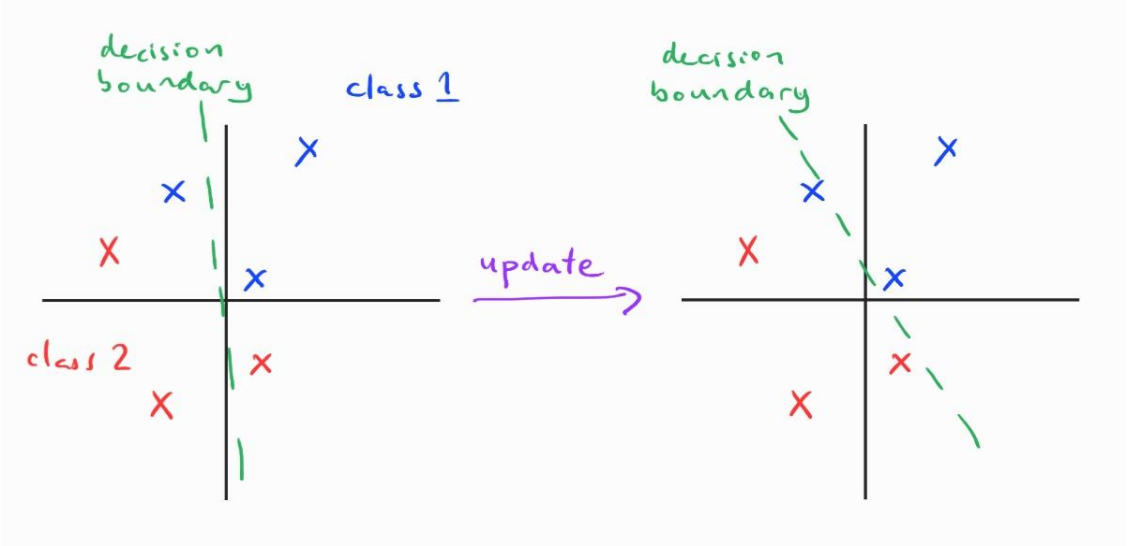


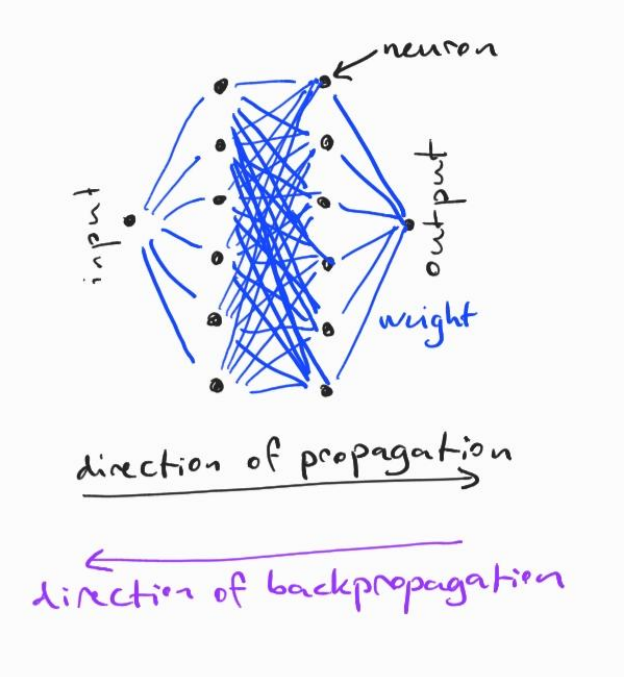
FIGURE 3
THE PROGRESS OF COMPUTING MEASURED IN COST PER COMPUTATION PER SECOND DEFLATED BY THE PRICE INDEX FOR GDP IN 2006 PRICES

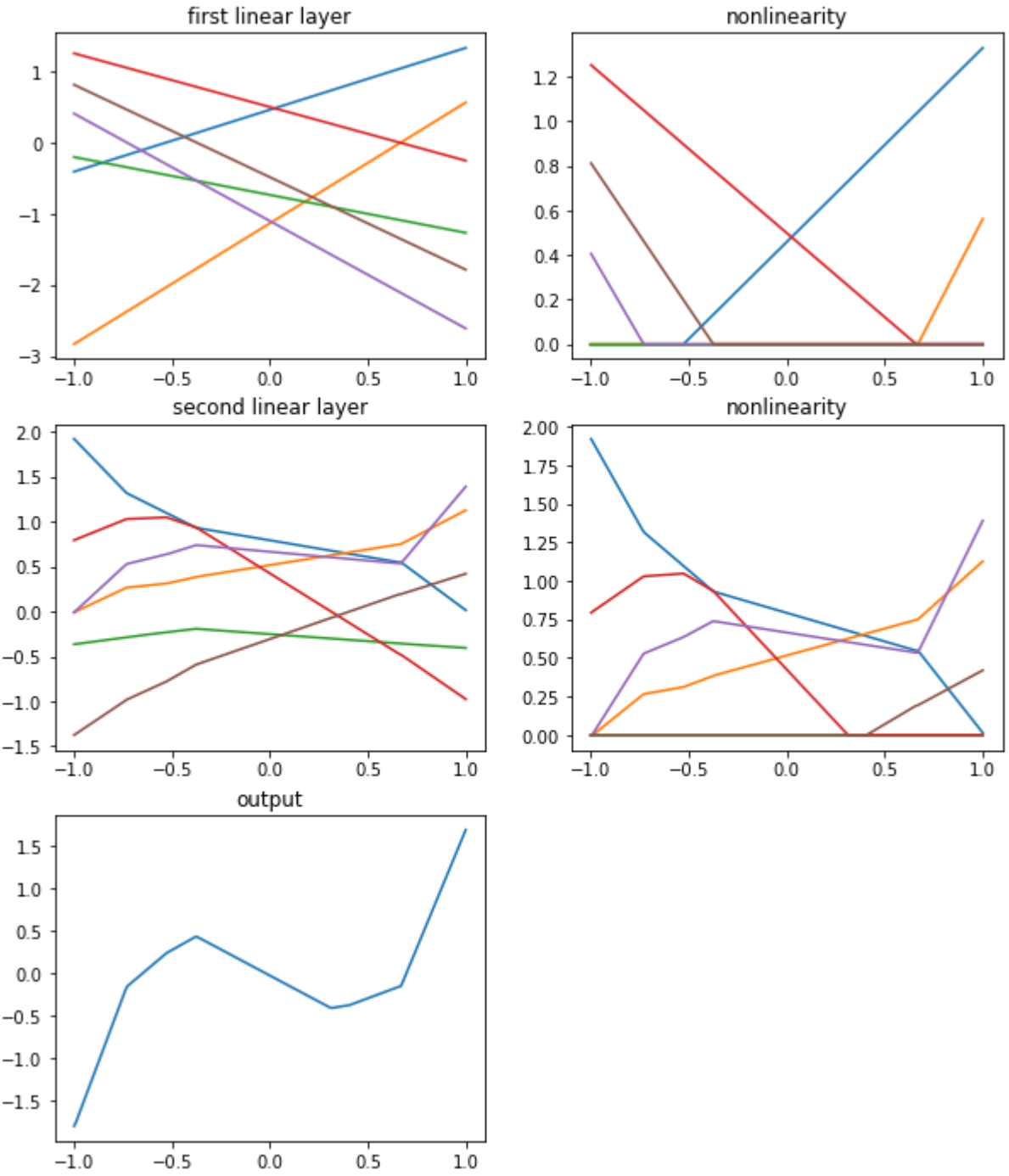


    - Especially good at hard-to-define tasks that work in high-dimensional spaces, e.g. classifying an image
    - Software 2.0: rather than programming a function (software 1.0), provide data that defines the input -> output pair
        - It turns out that for many interesting problems it's significantly easier to collect the data than to define the program
    - “Every time I fire a linguist, the performance of our speech recognition system goes up” (Fred Jelinek from 1985)
- Invention of transistor (1947) and Shannon’s theory of information (1948)
    - Events which kicked off the information age
- Perceptron (1957)
    - Linear model which is updated iteratively to find a decision boundary
    - Sketch of perceptron update:

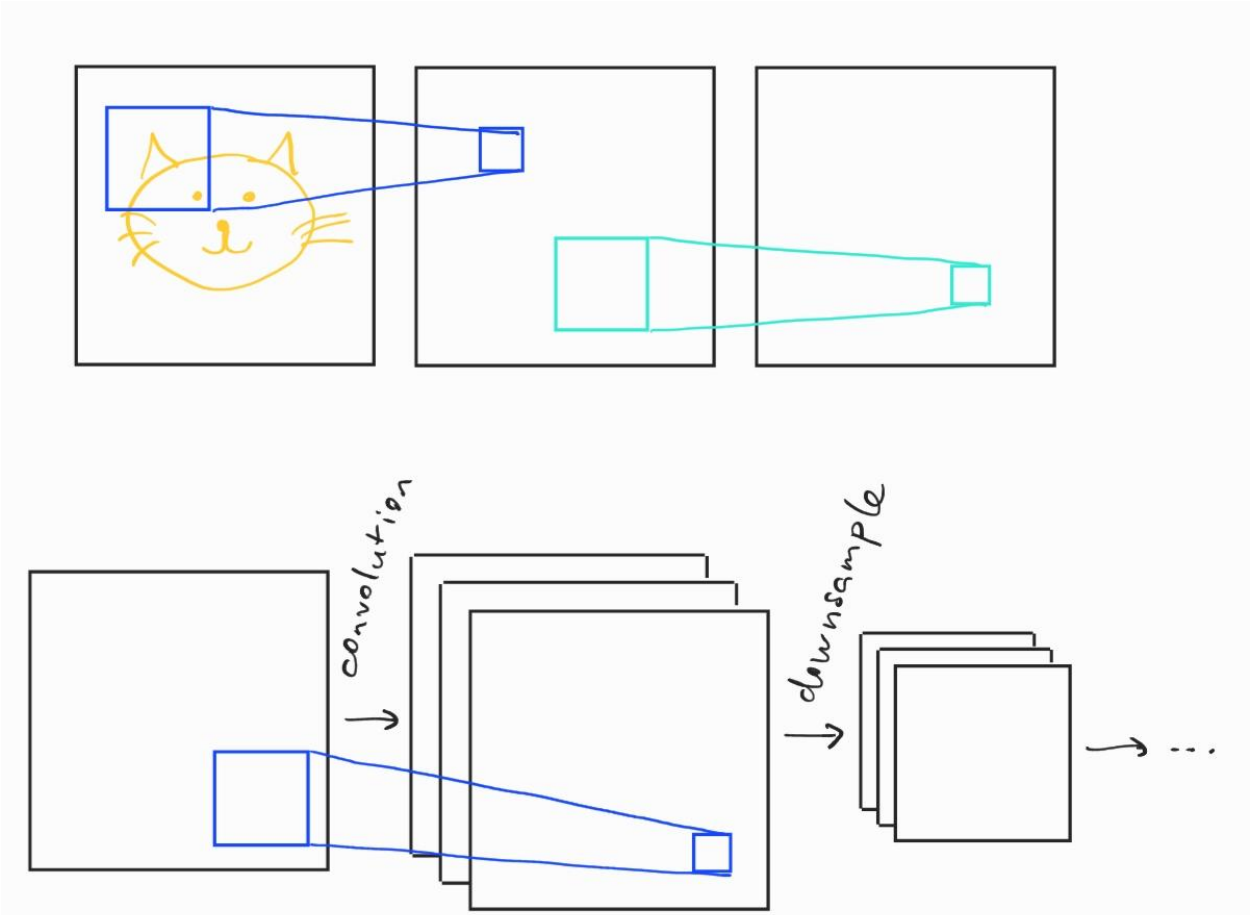


- Backpropagation (first described 1970, rediscovered 1986)
    - Allows training of multiple-layer neural networks
    - Example of three-layer network: 1D input to 1D output with 2 hidden layers of 6 neurons each

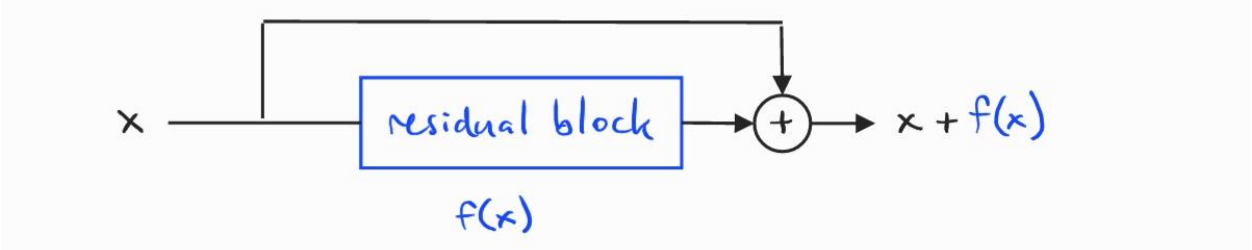




- Convolutional neural net (LeNet 1989)
    - Efficient image processing
    - Exploits translational symmetry of images
    - Sketch of stacked convolutional layers:

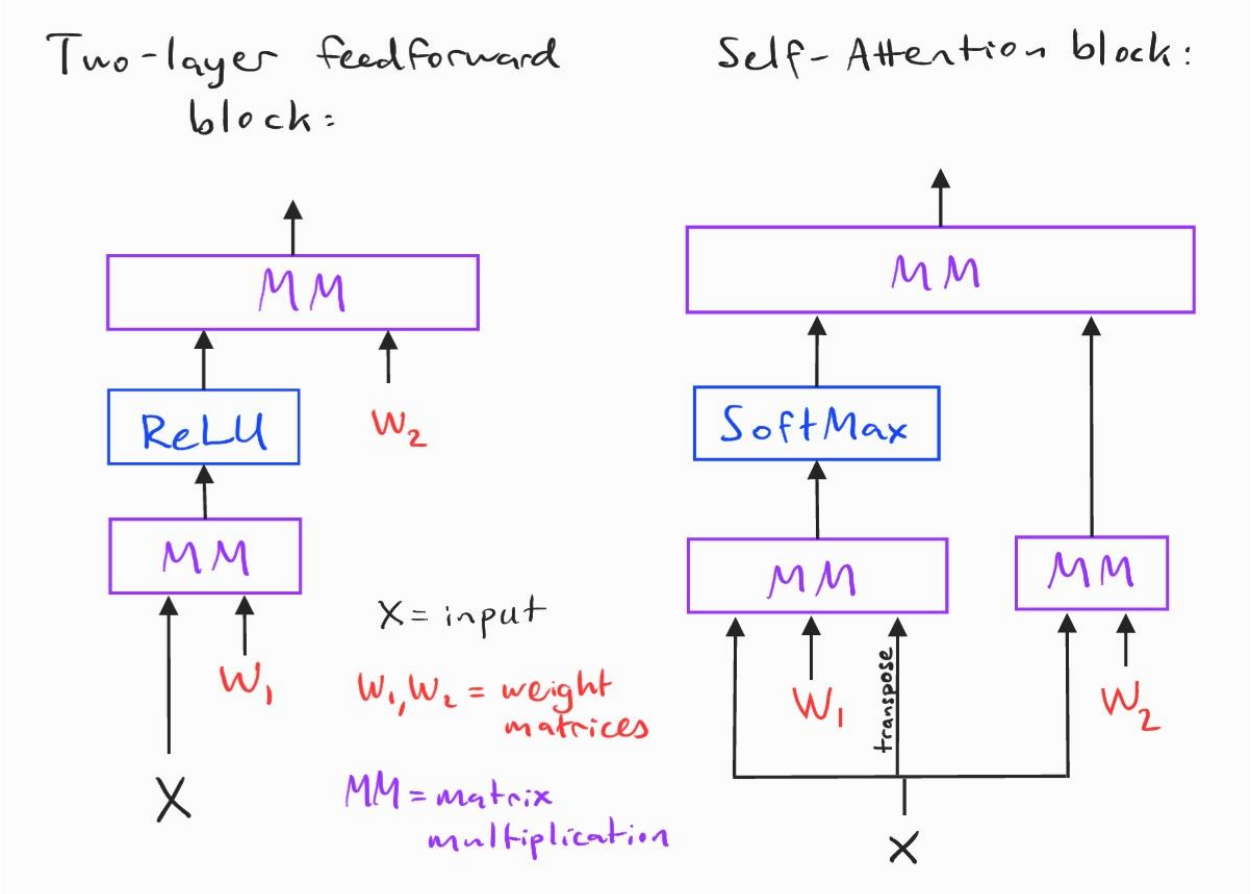


- Deep Blue beats Kasparov (1997)
- Development of GPUs (2000s)
- Deep neural nets (2010s)
    - ImageNet (2009)
        - Huge labeled dataset which kicked off deep learning
    - AlexNet (2012)
        - The model which made it clear that neural nets are the most powerful models around (at least for computer vision)
        - Used custom CUDA kernels to speed up convolutional nets
    - ResNet (2015)
        - Allowed very deep models, template for what followed
        - Sketch of residual blocks:



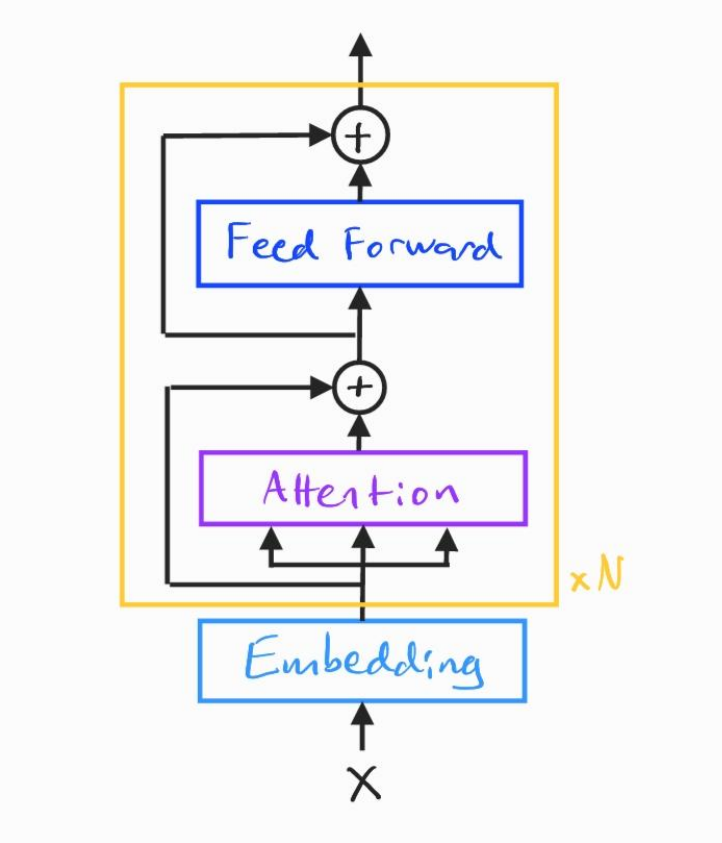
- AlphaGo (2016)
    - Maturation of reinforcement learning through self-play in discrete perfect-information games
    - AlphaZero a year later

- Transformer (2017)
    - General purpose model which can be applied to almost any type of data
    - Extensive use of attention mechanisms:



    - Attention is powerful because it allows for the expressive modeling of interactions within a sequence or set of data. Most data can be expressed as a sequence or a set (e.g. an image is a sequence of small patches)

    - Sketch of transformer structure:



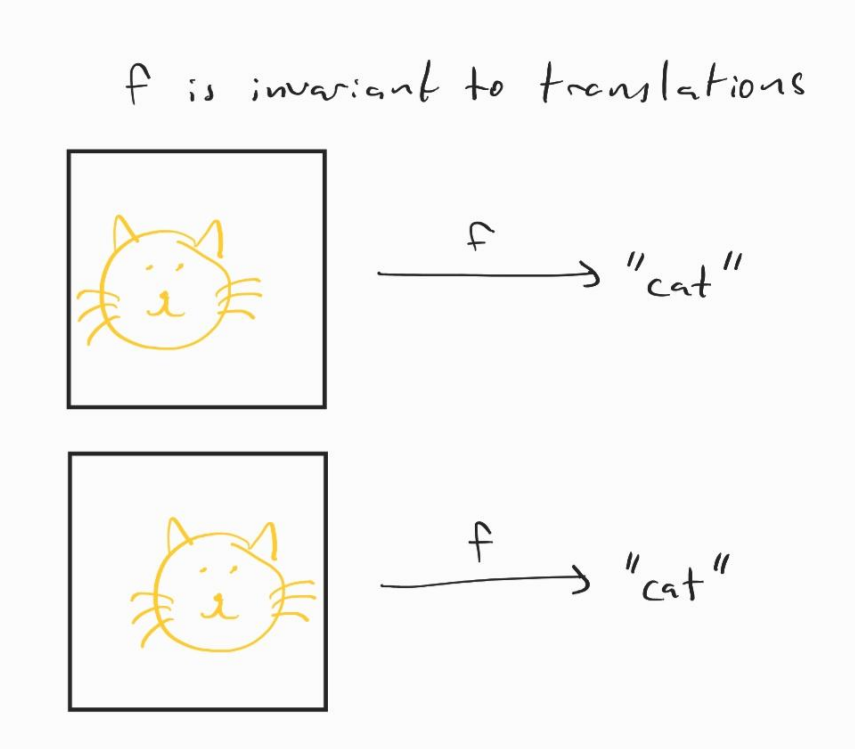
- GPT-3 (2020)
    - Large language model with surprisingly good fluency in natural language and ability to generalize and perform logical tasks
- AlphaFold 2 (2021)
    - Protein folding prediction to unprecedented accuracy
- The basics remain the same: sequential computations based on large matrix multiplications, trained via gradient descent on lots of data
- Interactive tool: see effect of NN layers and play around at https://playground.tensorflow.org/

Lecture 2

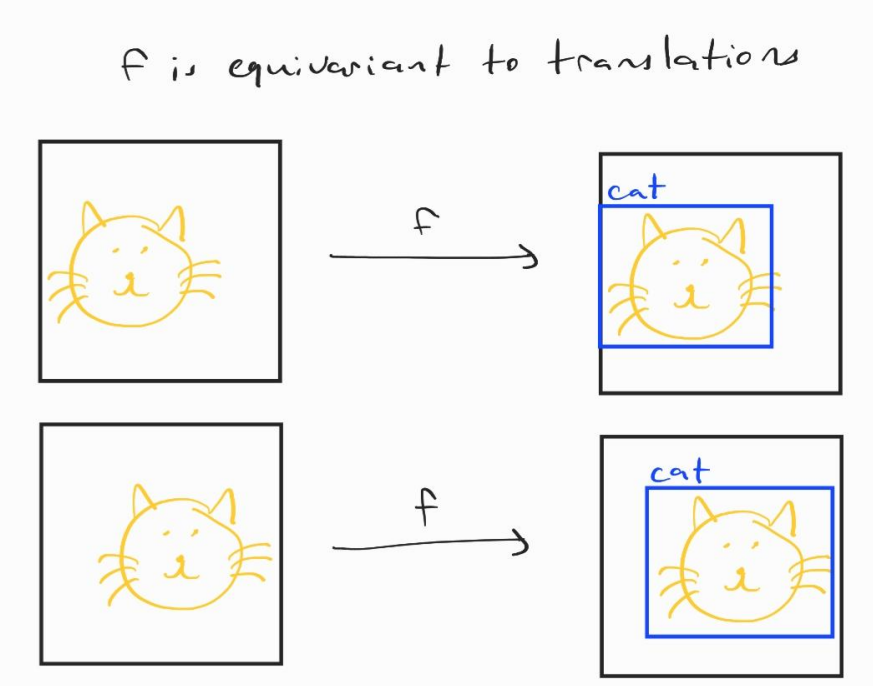
Applications to jet physics

Symmetries

- Symmetry is ubiquitous in physics
- In ML it is known that respecting the symmetries of the system in question means you need less data and compute
    - Convolutional neural nets (translational symmetry in images)
    - Set and graph networks (permutation symmetry)
    - Molecular modeling (isometries: translations, rotations)
    - If you have enough data and compute you can actually learn the symmetries, often ends up with better performance than hard-coding symmetries (e.g. ViT, i-GPT). But you need a lot of data and compute…
    - Can also learn to be invariant to symmetries using data augmentation and contrastive learning
- Difference between invariance and equivariance under symmetry
    - Invariance: the output doesn't change under a given symmetry (e.g. image classification)
        - Sketch:

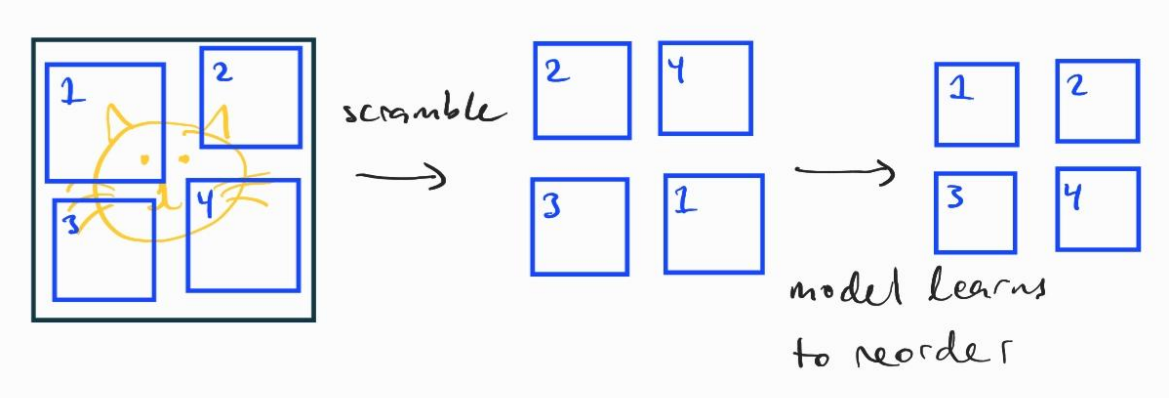


    - Equivariance: the output changes according to the same symmetry as the input (e.g. bounding boxes)
        - Sketch:



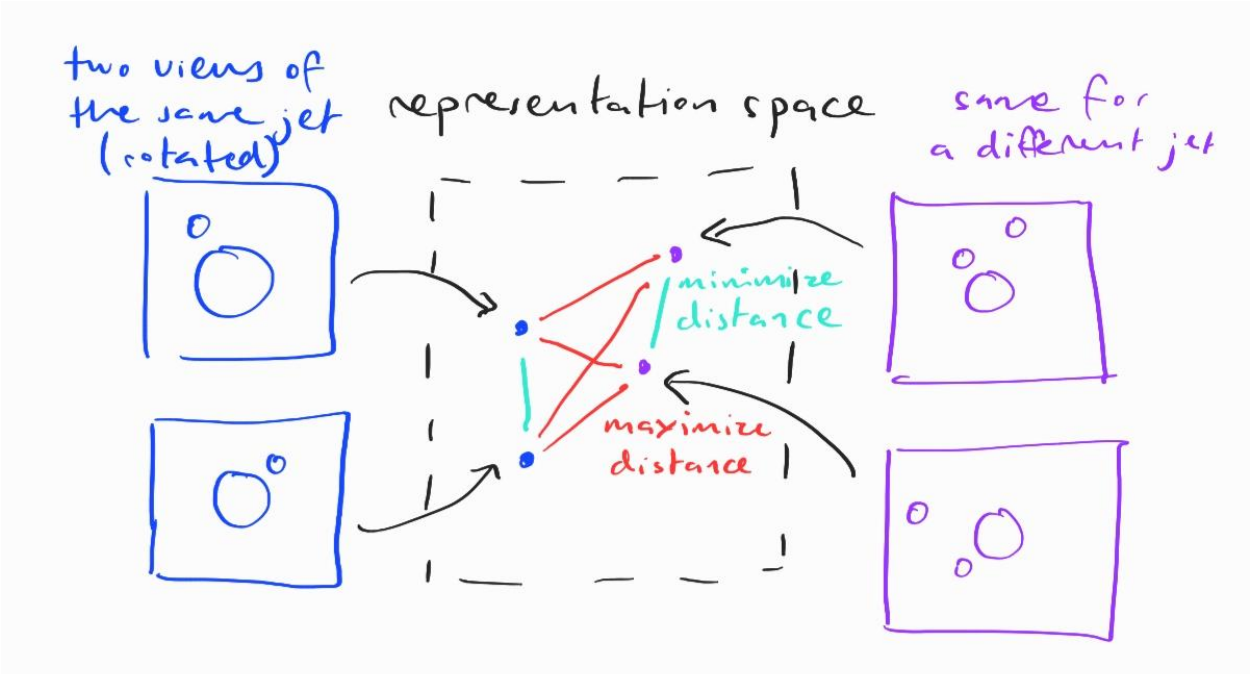
Representation learning

- You can take a classifier trained on a large labeled dataset, cut off the final layer and use it as a feature space to perform other tasks
- Is it possible to do the same thing without labeled training data? (labeled data is expensive to collect)
- Self-supervised learning (now one of the most important paradigms in deep learning due to the availability of unlabeled data)
- Create a task from data
    - Mask part of an image and predict the missing pixels
    - Predict the next word in a sequence of text
    - Jigsaw puzzle: cut an image into pieces, mix them up and learn how to reconstruct them. Sketch:

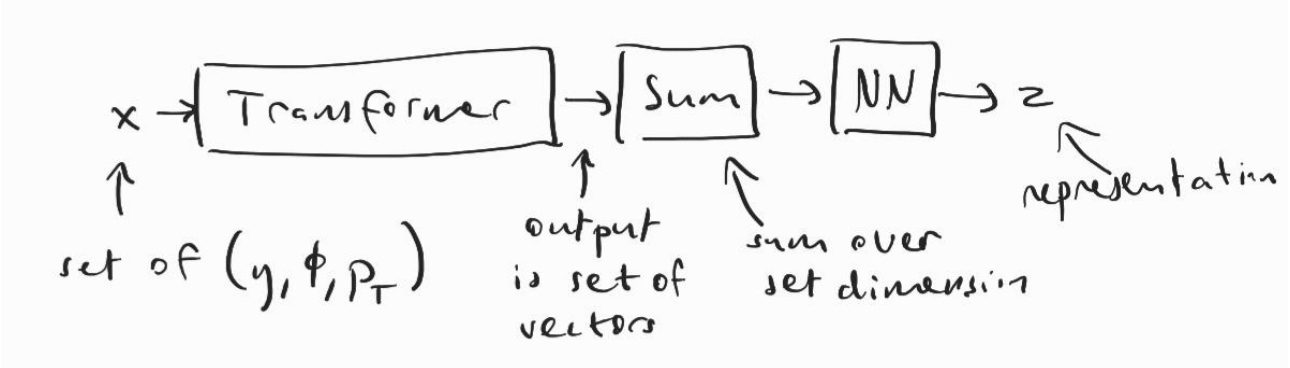


    - Rotate an image and learn to predict the rotation
- In order to complete the task the model has to extract non-trivial features from the data. The hope is that these features are useful for other tasks
- Type of self-supervised learning: contrastive learning

- Create pairs of data points (called augmentations) which represent the same underlying object
- E.g. images in which rotations, blurring, noise and other distortions added
- The model maps these pairs to a space (called the representation space) where they should be close together
- At the same time any representation should be far away from other representations which do not correspond to the same object
- Sketch:



- JetCLR: contrastive learning for jets
  - Jets represented by sets of (eta, phi, pT) coordinates
  - Augmentations defined by translations and rotations in (eta, phi) plane, addition of soft radiation, collinear splitting of particles
  - Augmentations define the learned invariances. Permutation invariance is hard-coded
  - Use a transformer (permutation equivariant) followed by a sum (permutation invariant) and feedforward nets to map to representation space. Sketch:

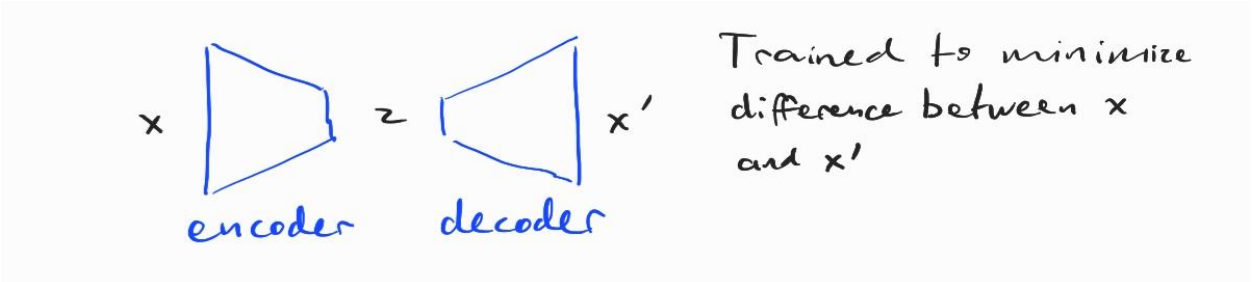


- JetCLR representations allow for a simple linear classifier to perform well on a supervised classification task (tag top quark jets in a QCD background)
  - Better than a hand-crafted baseline (energy flow polynomials)

    - Hope is that representations can be used for other downstream tasks, e.g. anomaly detection

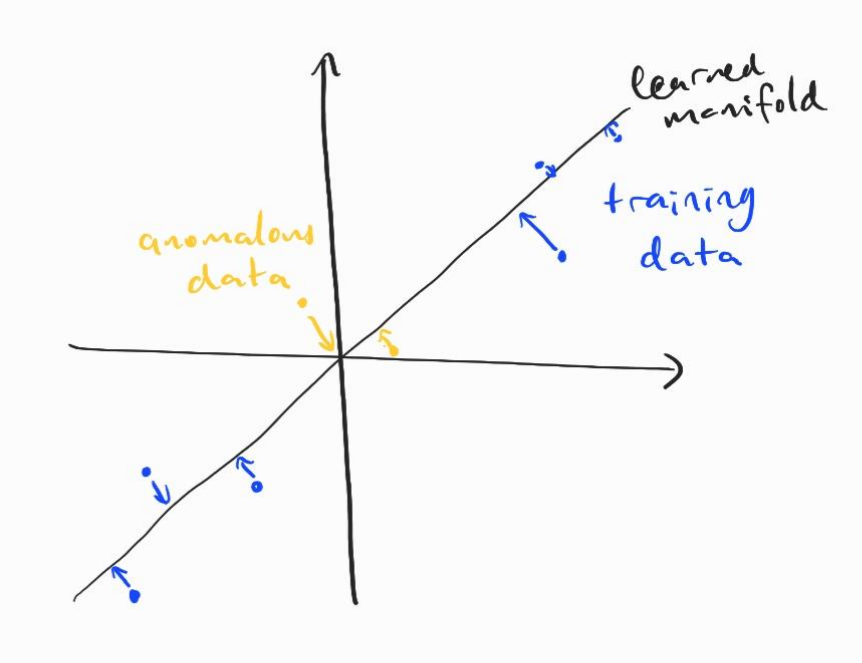
Anomaly detection

- At the LHC there is a huge stream of data. Rare events corresponding to new physics might be present in the data but haven't be found yet because the search space is too large
- Traditional methods are model based, e.g. testing a model hypothesis against the data, e.g. a bump hunt
- It would be useful to also have model-free methods, that can flag anomalies without making assumptions about the processes that generated them
- Proposal in jet physics: autoencoders
- Autoencoders try to reconstruct inputs while passing them through a bottleneck dimension. This forces the model to learn the most important features of the input and discard unneeded information. Sketch:



$$\mathcal{L} = \mathbb{E}_x[\|x - x'\|^2] = \langle \|x - x'\|^2 \rangle_x$$

- If we train autoencoders on normal, background data, we might be able to detect anomalies by looking for events with a high reconstruction error
    - This would suggest the features of the input are not similar to those of the training data, something expected of anomalies
- However, this doesn't always work as expected
    - If we train on QCD background, top jets are tagged as anomalous
    - But if we train on a top jet background, QCD jets are not tagged as anomalous, even though they are not seen during the training
    - The main problem seems to be that the autoencoder is still able to reconstruct signals of lower complexity than those it was trained on



- Solution (tentative): normalize the autoencoder
  - Don't just train the autoencoder to reconstruct inputs, but train it so that it fails to reconstruct inputs not in the training set

$$\mathcal{L} = \mathbb{E}_{x_+}[||x - x'||^2] - \mathbb{E}_{x_-}[||x - x'||^2]$$

  - This training method is more expensive but more effective
  - We can improve the ability of the autoencoder to detect QCD jets as anomalous when trained on top background
  - Also shows promise in detecting dark matter jets which have lower complexity than normal QCD background

Lecture 3

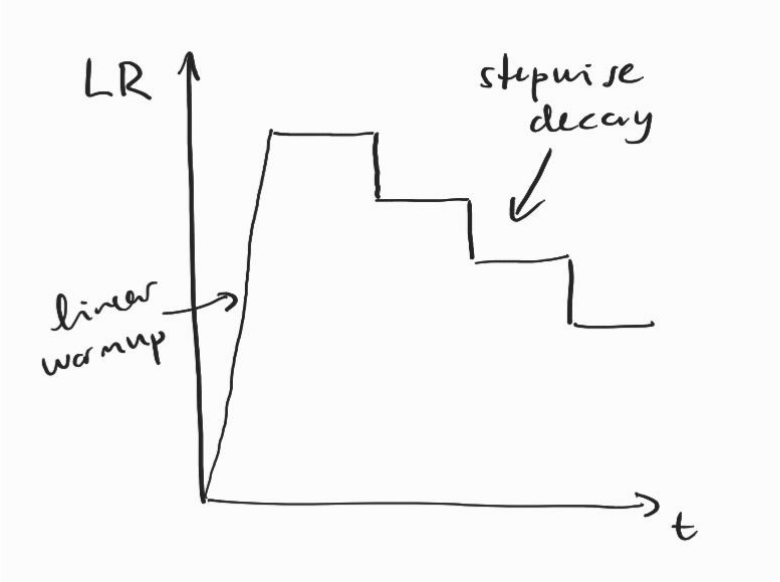
Tricks and trends

Practical tips for a successful ML project

- It's not easy to get ML working on a physics application out of the box. It takes time and effort for cutting-edge results
    - Budget six months for a PhD to complete a project, plus plenty of compute time (GPUs)
- It's more of an art than a science. There is no prescriptive formula for what will work. You need to know the tools available to you and try them out methodically
- However, there is a standard "bag of tricks" you can draw upon. Some combination of these will usually get your model working
    - Hyperparameter search: often most efficient to keep the number of free hyperparameters as low as possible and do a grid search
    - Regularization, particularly weight decay

$$\mathcal{L}_{\text{wd}} = \mathcal{L}_0 + \frac{1}{2}\|\theta\|^2$$

    - Normalizations, e.g. BatchNorm
    - Learning rate schedules: warmup, LR decay, cyclic learning rates



- The following also usually help but require more resources:
    - Increased model size
    - More training iterations
    - More data
    - Better data (e.g. less noisy)
- Tools available today which are of general scientific interest:
    - Surrogate models for simulations
    - Inverse models

Current trend: large language models

- Large language models: trained to predict the next word on a huge corpus of text
- Show remarkable, unexpected abilities, approaching human levels
    - Generating text in a specific style
    - Mathematical problem solving
    - Common sense reasoning
    - Explaining jokes
- Important application: Github Copilot code completion tool
    - Best feature: give explanation of the behavior of a function as a comment and Copilot can write the function for you (or at least will try)
    - For physics PhDs: could greatly speed up the ability to work with legacy codebases passed down from student to student (a common and very time-consuming problem)
- These models are interesting because it shows how flexible language is as an interface with ML models
    - Example: Flamingo is a text and image model which can interactively answer questions about the contents of an image.
    - Similar models for less human-readable forms of data could become very important in data exploration and analysis
- In the future we will possibly see natural language become the standard interface for interacting with machine learning models